UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESTUDIOS BÁSICOS

DEPARTAMENTO DE COMPUTACIÓN

DESARROLLO DE UNA BIBLIOTECA DE CLASES EN JAVA PARA RESOLVER PROBLEMAS DE COMPUTACIÓN Y ÁLGEBRA MATRICIAL MEDIANTE EL USO DE ARREGLOS BIDIMENSIONALES.

Trabajo de Investigación presentado como requisito para optar a la categoría de Profesor Asistente

AUTOR: Bolívar P., Alejandro E.

TUTOR: Prof. Mayela Delgado

Valencia, Febrero de 2013.

ÍNDICE GENERAL

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESTUDIOS BÁSICOS

DEPARTAMENTO DE COMPUTACIÓN

AUTOR: Bolívar P., Alejandro E.

Fecha: Febrero de 2013.

RESUMEN

La Universidad es fuente inagotable de producción de conocimiento para el desarrollo de la ciencia y la tecnología en el país, el diseño y creación de aplicaciones del área de computación y matemática requieren de una gran dedicación para el diseño, edición, simplificación, validación, y reutilización de código para la resolución de problemas que por su complejidad y extensión requieren de bibliotecas de métodos para facilitar la creación y mantenimiento de paquetes de aplicaciones con un mayor alcance en el área de estudio. Este trabajo tiene como objetivo, desarrollar una biblioteca de clases en java, para resolver problemas de computación y álgebra matricial mediante el uso de arreglos bidimensionales, que atienda a las necesidades de los docentes, investigadores y estudiantes de la Facultad de Ingeniería de la Universidad de Carabobo, como una ayuda para el desarrollo de aplicaciones tomando como base el empleo de arreglos bidimensionales. Los rasgos importantes se determinaron a partir de fuentes tanto primarias como secundarias, que incluyeron, la observación directa y la aplicación de un instrumento de recolección de datos. Se utilizó un cuestionario estructurado con escala tipo Likert, para evaluar los aspectos que definen a la percepción que tiene el estudiante acerca de la implementación de los arreglos para la resolución de problemas prácticos de la ingeniería y la opinión de expertos en el área. Finalmente, esta investigación, busca definir las actividades básicas de programación de uso en la ingeniería para que la biblioteca, se pueda integrar como una herramienta de colaboración en la enseñanza por parte del profesorado y adicionalmente provea facilidades para futuros trabajos de investigación.

Palabras clave: Biblioteca de métodos, subprogramas, funciones, arreglos, matrices.

INTRODUCCIÓN

Las operaciones matriciales tienen una amplia aplicación para resolver problemas en distintas áreas del conocimiento; entre ellas la ingeniería en sus diversas ramas como los son: los métodos numéricos, la estadística, la economía y muchas otras más; así como también en el área de computación. Su aplicación alcanza el estudio del álgebra matricial, operaciones elementales matriciales, la factorización y resolución de sistemas de ecuaciones lineales; todo esto con el fin de resolver redes de fluido o de circuitos, problemas de transporte, tratamiento de imágenes, entre muchos otros estudios. También se emplea para diversas operaciones en el trabajo relacionado con matrices, las matrices booleanas, la actualización, búsqueda y ordenamiento en matrices; todas estas actividades requieren de procedimientos básicos de creación e identificación de diferentes tipos de matrices.

En este trabajo se plantea la creación de una biblioteca de clases para resolver problemas del área de ingeniería y de computación, dicha biblioteca está desarrollada en el lenguaje de programación Java utilizando el editor NetBeans IDE 7.2.1, este lenguaje es de amplio uso actualmente tanto en Windows como en Software Libre, como lo indica el Índice de la Comunidad de Programación en febrero de 2013 (TIOBE, 2013), donde se muestra las estadísticas del índice de popularidad de los lenguaje de programación más utilizados en la actualidad, TIOBE es un indicador de la popularidad de los lenguajes de programación, el índice se actualiza una vez al mes, las calificaciones se basan en un gran número de ingenieros expertos de todo el mundo, cursos y proveedores; los motores de búsqueda populares de Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube y Baidu se utilizan para calcular las calificaciones.

Las bibliotecas de programas son fundamentales para incrementar la eficiencia de transcripción de código, profundización y ampliación de las aplicaciones a desarrollar además de facilitar el mantenimiento y mejoras de las aplicaciones ya existentes que las utilicen.

Este trabajo está conformado por cinco capítulos, en el primer capítulo se describe el planteamiento del problema; así como también, el objetivo general y específicos, de la misma manera se expone la justificación, y alcance de la investigación. El segundo capítulo tiene los antecedentes relacionados con la investigación y el material teórico de apoyo a la investigación. En el tercer capítulo se define el tipo y diseño de la investigación, así como también la descripción de todos los elementos relacionados con las diferentes etapas o fases del proceso metodológico. En el cuarto capítulo se encuentra la discusión y análisis de los resultados obtenidos y, se desarrolla la propuesta de la investigación conformada por el conjunto de clases de la biblioteca para resolver problemas de algebra matricial y de computación. Finalmente en el quinto capítulo se presentan las conclusiones y recomendaciones de la investigación y se presentan las referencias citadas y los anexos incluyendo un CD con el código fuente de la biblioteca que es el producto de la investigación, incluye también el archivo .jar y la documentación de toda la biblioteca.

CAPÍTULO I

EL PROBLEMA

En este capítulo se presenta el planteamiento del problema, especificando la formulación del mismo, así como también los objetivos planteados para cumplir con la realización del presente trabajo, mostrando además, la justificación que conlleva a la ejecución de la investigación y el alcance.

Planteamiento del problema

El desarrollo de la tecnología avanza a grandes pasos, la sistematización de los procesos tanto en la industria como en el comercio y los nuevos equipos tecnológicos presentes en la vida cotidiana, la medicina, telecomunicaciones, y en otras actividades, como la práctica educativa plantean realizar cambios significativos que colaboren y fortalezcan el proceso de enseñanza y aprendizaje en el sector educativo del país.

El diseño y elaboración de un proyecto de investigación que requiere como apoyo la elaboración de un programa en computadora, depende inicialmente de la búsqueda de código fuente ya creado en el mismo lenguaje o en su defecto en cualquier otro lenguaje; incluso un algoritmo, pseudocódigo o diagrama de flujo, para posteriormente integrarlos y obtener la aplicación final. Debido a esto es recomendable disponer del código necesario, debidamente documentado para facilitar el trabajo de programación y favorecer la posibilidad de tener un mayor alcance en el programa a desarrollar, ya que el tiempo a invertir en el desarrollo de ese conjunto de procedimientos, muchas veces de nivel básico, se puede emplear precisamente en el abordaje del programa de manera avanzada.

Las bibliotecas de programas son parte fundamental en la creación de las aplicaciones, y para los desarrolladores de programas en el área de investigación, académica, industrial y empresarial, son herramientas de programación base para el mejoramiento de programas y evitan: la duplicación de código, además de la pérdida de tiempo en la búsqueda y modificación o adaptación de código.

Las bibliotecas están desarrolladas en lenguajes de programación tales como: Matlab, C, Fortran, Java entre otros; esto implica que si se consigue el código fuente en otro lenguaje diferente al que el investigador o estudiante domina o conoce, debe realizar un esfuerzo adicional en función de interpretar el código y aún más si el lenguaje de programación utiliza funciones intrínsecas para realizar alguna actividad; pues entonces se tendrá que desarrollar dicho código. La biblioteca también se puede encontrar empaquetada como: .dll, .api, .jar entre otras, lo que impide conocer la estructura y los procesos que se realizan internamente.

En internet se puede conseguir sitios que contengan bibliotecas o segmentos de código que se pueden utilizar para realizar alguna actividad, sin embargo son susceptibles a las desventajas propias de toda información presente en internet como lo son: presencia de virus, disponibilidad de conexión con el servidor, si es de algún lugar remoto y no se tiene información directa para contactar al propietario, se dificulta la obtención del recurso requerido.

En el caso particular de la resolución de sistemas de ecuaciones lineales, existen bibliotecas que implementan diferentes métodos de resolución con técnicas avanzadas de programación como en el caso de las matrices densas o esparcidas, dichas matrices son convertidas en un formato adecuado (CSR, DIA, COO, ELLPACK…) según sea el caso para resolver el sistema; el estudio de este tipo de matrices no se contempla en la carrera de ingeniería; dicha biblioteca resuelve el sistema de ecuaciones, sin embargo implementa técnicas de resolución desconocidas para los estudiantes de pregrado.

De la misma manera, las matrices son de amplio uso para la resolución de problemas en ingeniería, así se refleja en un artículo para el estudio de la rehabilitación de estructuras de hormigón mediante aplicaciones matriciales (Almeida Benítez, Márquez Rodríguez, & Franco Brañas, 2001). Los autores consideran un ejemplo que puede ser utilizado a nivel universitario considerando algunos tópicos interesantes, tales como ecuaciones en diferencias, diagonalización de matrices, potencia de una matriz, cadenas de Markov, etc. Este artículo manifiesta la necesidad de la implementación de matrices para la resolución de ejercicios prácticos y la diversidad de operaciones algebraicas que se deben utilizar.

En el libro "Matemáticas discretas para la ciencia de la computación" se señala que las matrices se utilizan en el cálculo numérico, en la resolución de sistemas de ecuaciones lineales, de las ecuaciones diferenciales y de las derivadas parciales. Además de su utilidad para el estudio de sistemas de ecuaciones lineales, las matrices aparecen de forma natural en geometría, estadística, economía, informática, física, y muchas otras más (Calderon Vilca, 2008).

En la Facultad de Ingeniería de la Universidad de Carabobo, no se dispone de software educativo ni de una biblioteca de códigos para resolver problemas de álgebra matricial en Java, por lo que se debe recurrir a libros o buscar en internet para descargar alguna que se ajuste lo más cercano a las necesidades del proyecto a desarrollar; bien sea del mismo lenguaje, alguno parecido o en caso extremo el algoritmo o diagrama de flujo. Esto hace que se tenga que emplear un tiempo considerable, en el caso de encontrar alguna biblioteca alternativa, para revisar cuales métodos tiene, el tipo de variables y tipo de dato que utiliza, entre otras verificaciones. Lo expuesto anteriormente conlleva a la necesidad de integrar y consolidar en una biblioteca los métodos relacionados con el cálculo matricial y las operaciones del área de computación para implementar en trabajos de investigación y en la resolución de problemas en el área de ingeniería.

El estudiante de ingeniería en el segundo semestre cursa estudios básicos del álgebra lineal e inicia el estudio de resolución de problemas mediante el uso de arreglos cuando cursa la asignatura Computación II, a excepción de ingeniería mecánica; sin embargo es en Computación Avanzada, asignatura electiva del departamento de computación ofrecida a todas las escuelas de ingeniería excepto civil, cuando se implementa el tema mediante el lenguaje de programación Java; en dicha asignatura es necesario estudiar con mayor profundidad el uso de las matrices para su aplicación en el área de computación y de ingeniería, por lo tanto se requiere implementar diversas aplicaciones procurando emplear un menor tiempo de edición de código en la clase.

Entre las dificultades que se les presentan a los alumnos, se tienen:

1. La repetición de código.
2. La falta de tiempo para la transcripción del código en horas de clase, y consecuentemente la imposibilidad de realizar comparaciones entre diferentes métodos de resolución para algún caso en particular.
3. La falta de disponibilidad de un conjunto de programas, para desarrollar diversas actividades, que reflejen la importancia de la implementación de las matrices en la resolución de problemas en el área de computación y de ingeniería, que apoyen el proceso de enseñanza y aprendizaje.

En semestres posteriores el estudiante utiliza las matrices para la resolución de ejercicios en asignaturas como: métodos numéricos, mecánica racional, manejo de fluidos, transferencia de calor, análisis de esfuerzo, teoría de control III, entre otras asignaturas dependiendo de la carrera; cabe destacar que estas materias no cuentan con software educativos para el cálculo o simulación de los casos de estudio, sino que se limitan a la resolución de problemas con sistemas de ecuaciones de pequeñas dimensiones 3 x 3 o 4 x 4, utilizando objetos de aprendizaje como presentaciones, páginas web, aulas virtuales y documentos de texto.

Finalmente como base legal la Constitución de la República Bolivariana de Venezuela de 1999 en su preámbulo y en los artículos 108 y 110 presenta la importancia que tiene para el estado, la implementación y desarrollo de las nuevas tecnologías con el fin de fortalecer el proceso educativo en los centros educativos y para el desarrollo del sector económico y social del país. Actualmente el Ministerio del Poder Popular para la Educación Universitaria, mediante el Programa Fomento a la Educación Universitaria (ProFE) está desarrollando cursos virtuales de calidad en diferentes áreas para las universidades oficiales, en las áreas de computación, procesos químicos, mecánica, electricidad, construcción civil, entre otras áreas (Fuente: http://ead.opsu.gob.ve/joomla/moodle19/moodle/course/view.php?id=34). En vista de esta necesidad, esta investigación empleada como software educativo permitirá a nivel nacional fortalecer la enseñanza del uso de los arreglos bidimensionales a nivel presencial, semi presencial y a distancia.

Formulación del Problema

Para orientar la investigación, cuyo propósito es ofrecer a la comunidad académica de la Facultad de Ingeniería de la Universidad de Carabobo y a los desarrolladores de programas en general, un software educativo como soporte a las investigaciones, trabajos de grado y la enseñanza, se debe dar respuesta a la siguiente interrogante: ¿Cuáles métodos han de considerarse para crear e integrarlos en una biblioteca de clases en java para resolver problemas de computación y álgebra matricial, mediante el uso de arreglos bidimensionales?

Objetivos

Objetivo general

Desarrollar una biblioteca de clases en java, para resolver problemas de computación y álgebra matricial, mediante el uso de arreglos bidimensionales.

Objetivo específicos

Con la finalidad de lograr el alcance del objetivo propuesto, se plantean los siguientes objetivos específicos:

- Diagnosticar, mediante un proceso de levantamiento de información, las necesidades esenciales de implementación de matrices en la resolución de problemas, en el área de computación e ingeniería.
- Diseñar mediante un modelo orientado a objetos, la estructura, clases y métodos de la biblioteca para la creación, identificación, cálculos y operaciones matriciales que se emplean en el área de computación e ingeniería.
- Crear la biblioteca para el manejo de arreglo bidimensional, utilizando las técnicas de manejo de matrices apropiadas.
- Realizar pruebas de funcionamiento de la biblioteca, basadas en su integración y uso en una aplicación para el área de ingeniería.

Justificación

La biblioteca contiene las operaciones básicas de matrices (suma, producto, etc.), identificación de matrices (identidad, simétrica, booleana, y otras), matriz inversa, determinantes, factorización, resolución de sistemas de ecuaciones, actualización (intercambiar filas/columnas, ampliar matrices, entre otras operaciones) cuyas definiciones están dentro del área de matemática; el álgebra lineal, los métodos numéricos, la matemática discreta, las ecuaciones diferenciales entre muchas otras, contemplan el estudio de métodos y técnicas de cálculo para resolver problemas mediante el empleo de matrices.

El desarrollo tecnológico incide en las aulas de clase, esto ofrece al docente la posibilidad de ofrecer al estudiante alternativas diferentes de aprendizaje para abordar temas de mayor complejidad que en una clase tradicional difícilmente se pueda desarrollar, tal es el caso de resolver ejercicios prácticos como por ejemplo, el análisis

del efecto de la variación del caudal externo en una red de tuberías o realizar la comparación de la ejecución de diferentes métodos de ordenamiento o de búsqueda.

Por otro lado, las actividades de búsqueda, ordenamiento y las operaciones elementales en matrices, etc. se pueden utilizar en actividades como: la automatización de procesos administrativos, del personal, control de datos de estudiantes para las dependencias, ordenar las matrices para verificar si es una matriz triangular y resolver directamente por sustitución.

En el ámbito académico tendrá un impacto positivo su implementación ya que se benefician docentes, investigadores y estudiantes. El producto de esta investigación fortalecería la producción de futuras investigaciones en el área de ingeniería, con el desarrollo de dichas investigaciones, o tomando como base este trabajo también se beneficiaría las industrias y empresas a las cuales se le desarrollan trabajos de grado; de la misma manera con la comunidad, ya que los programas a desarrollar en los trabajos de grado o servicio comunitario, han de ayudar a resolver los problemas prácticos que se puedan presentar en dichos sitios.

Para el docente, la biblioteca proporciona los métodos básicos para la producción de software educativo relacionado con la resolución de problemas mediante el uso de matrices. Por ejemplo: se puede diseñar aplicaciones o páginas web para visualizar paso a paso los métodos de factorización o de resolución de sistemas de ecuaciones lineales, incluyendo la posibilidad de interactuar con el usuario; de manera similar se puede trabajar con temas como: operaciones matriciales, identificación de matrices, búsqueda y ordenamiento de datos.

Adicionalmente, se puede también crear simuladores para ejemplificar en asignaturas más avanzadas, procesos de cálculo como: circuitos eléctricos, armaduras, redes de fluido, cálculo de esfuerzos y deformaciones, operaciones básicas de computación gráfica, actividades lúdicas, entre otras aplicaciones potencialmente beneficiosas para el proceso de enseñanza y aprendizaje.

En la asignatura Computación Avanzada, aporta a los estudiantes de manera significativa un apoyo académico para el estudio y desarrollo de los temas como: constructores, métodos, arreglos bidimensionales, búsqueda y ordenamiento, entre otros. Además, de la elaboración de proyectos mediante la implementación de la biblioteca de métodos, en este caso particular orientado a las operaciones mediante el álgebra matricial y de computación. Adicionalmente, se incentiva a la comunidad estudiantil a la implementación de la biblioteca, para resolver ejercicios de otras asignaturas.

Alcance de la investigación

La investigación será una herramienta de apoyo a desarrolladores de software educativo, para implementar en cualquier asignatura que requiera de la resolución de problemas mediante el uso de arreglos bidimensionales, como por ejemplo: álgebra lineal, ecuaciones diferenciales, métodos numéricos, mecánica racional I, manejo de fluidos, transferencia de calor, procesos de manufactura, sistemas de transporte, teoría de control III y otras asignaturas.

Las clases desarrolladas permiten la creación de matrices de tipo rectangular y cuadrada. Se puede crear la matriz desde un archivo o por teclado, también mediante operaciones para obtener una submatriz, la opuesta, de valores aleatorio, escalar, diagonal entre otras matrices.

La biblioteca abarca la identificación o clasificación de una matriz, identifica el tipo de dato, si está escalonada, es booleana, ordenada, diagonal, tridiagonal, ortogonal entre otros métodos de clasificación.

En la clase booleana, se realizan operaciones básicas para obtener la unión, intersección, complementaria y diferencia simétrica de matrices booleanas.

Para las operaciones matrices se obtiene como resultado un número escalar mediante las operaciones de suma, promedio, máximo, mínimo, determinante, etc.

También el resultado puede ser una matriz mediante operaciones como: suma, producto, traspuesta, inversa, entre otras operaciones.

Adicionalmente en lo que respecta a la resolución de sistemas de ecuaciones lineales, para lo cual se utilizan los métodos de la clase resolver, factorizar y actualizar, para implementar los métodos de resolución directo: Gauss, Gauss Jordan, Doolitle, Crout, Cholesky, Thomas, y los métodos iterativos: Jacobi y Gauss Seidel.

En el área de computación se incluye la búsqueda secuencial para datos ordenados y desordenados, la búsqueda binaria, alrededor de una celda, y de una submatriz dentro de una matriz. Los métodos de ordenamiento implementados son: iterativos como burbuja, inserción, selección, Shell y recursivo como quicksort.

Para la impresión por consola, el método *Salida* muestra toda la matriz o alguna parte en específico: el triángulo superior o inferior a la diagonal principal o secundaria, y la diagonal principal o secundaria. Esto es ideal para la enseñanza básica en álgebra lineal y por supuesto la visualización de la matriz como resultado de una operación.

CAPÍTULO II

MARCO TEÓRICO

En este capítulo se exponen los fundamentos que desde el punto de vista teórico sustentan el presente trabajo; estos se han dividido en dos partes: la primera, contiene una recopilación de algunas investigaciones realizadas en esta área de estudio, que bien sea por su contenido o metodología servirán de base para el desarrollo de la investigación. La segunda se refiere a las bases teóricas, en las cuales se presentan los principios sobre los cuales se sustenta el trabajo.

Antecedentes de la investigación

La creación de bibliotecas de programas es una práctica comúnmente realizada por los desarrolladores de software, con la finalidad de tener a la mano segmentos de código que faciliten la edición de los programas y así evitar la repetición de código ya creado, para el manejo de arreglos existen en su mayoría bibliotecas desarrolladas en Fortran o en Lenguaje C, lógicamente también en Java existen algunas bibliotecas desarrolladas por institutos de investigación y grupos de programación, las cuales se enfocan en casos particulares según la investigación que estén desarrollando, entre ellas se tiene a *Colt, Commons Math, Efficient Java Matrix Library (EJML), Jama, jblas, JScience (Older benchmarks only), Matrix Toolkit Java (MTJ), OjAlgo, Parallel Colt, Universal Java Matrix Package (UJMP), Elegant Linear Algebra for java (la4j)*. En este conjunto de bibliotecas se observa en común el estándar o convenciones de programación de las clases y métodos en Java; estas bibliotecas están orientadas al trabajo con matrices densas y dispersas, implementando técnicas y métodos avanzados para resolver sistemas de ecuaciones.

En el trabajo “Introducción al uso de bibliotecas de álgebra para estudiantes de ingeniería” se expone fundamentalmente la existencia de muchas bibliotecas para Java, y que son de uso libre y gratuito (Di Mare, 2010). En este trabajo se explica cómo lograr que un estudiante utilice la biblioteca JAMA de manipulación de matrices, pues después que el alumno aprende a incorporar una biblioteca en su programa puede utilizar otras más, según las necesite. La elección de la biblioteca JAMA, producida por el *“National Institute of Standards and Technology”*, se fundamenta en estas cualidades:

- Es bastante completa, pues incluye la mayor parte de las operaciones en matrices que se necesita para resolver problemas prácticos.
- Como es simple sirve para introducir los conceptos inmediatamente.
- Incluye el código fuente lo que facilita entender cómo funciona la biblioteca.
- Está construida por una organización de buen prestigio en USA y en el mundo

JAMA se compone de seis clases en Java: Matrices, descomposición Cholesky, LU, QR, Valor Singular y Eigenvalores.

Por otro lado, la biblioteca JAMA no abarca la identificación del tipo de matriz, la búsqueda de un valor en la matriz tanto por fila como por columna, tampoco contiene métodos para el ordenamiento de la matriz por fila o por columna.

El paquete de matriz Universal de Java, conocida por sus siglas en inglés como (UJMP) es una biblioteca de Java que proporciona implementaciones para matriz dispersa y densa, así como cálculos de álgebra lineal, como la descomposición de la matriz, inversa, multiplicación, media, correlación, desviación estándar, etc.

La biblioteca la4j (álgebra lineal para Java) de Java es elegante y pura para resolver problemas de álgebra lineal, sirve de apoyo para vectores y matriz dispersa y densa.

La biblioteca de matriz de Java eficiente (EJML) es una biblioteca de álgebra lineal para manejar matrices densas. Los objetivos de diseño son: 1) es lo más eficiente posible computacionalmente y en memoria para matrices grandes y pequeñas y 2) es accesible para principiantes y expertos. Estos objetivos se realizan seleccionando los mejores algoritmos para utilizar en tiempo de ejecución y diseño de una API limpia. EJML es gratuito, escrito en Java 100% y ha sido publicado bajo una licencia LGPL, proporcionando la siguiente funcionalidad:

- Operadores básicos (suma, multiplicación, ...)
- Manejo de matrices (extraer, insertar, combinar, ...)
- Soluciones lineales (lineal, mínimos cuadrados, incremental, ...)
- Descomposición (LU, QR, Cholesky, SVD, Eigenvalue, ...)
- Características de la matriz (rango, simétrica, ...)
- Matrices aleatorias (covarianza, ortogonal, simétrica, ...)
- Formatos internos diferentes (fila mayor, bloque)

Estas tres últimas bibliotecas implementan métodos de resolución de sistemas de ecuaciones lineales cuyo estudio no se contempla en la carrera de ingeniería, por lo tanto no se implementan a nivel de pregrado.

La interfaz *RealMatrix* representa una matriz de números reales como entradas, se admiten las siguientes operaciones de matriz básica:

- Suma, substracción y multiplicación de matrices.
- Suma y multiplicación escalar.
- Traspuesta
- Traza
- Operaciones en vectores.

La utilización de matrices (*arrays*) constituye actualmente una parte esencial en los lenguajes de programación, ya que la mayoría de los datos se introducen en los

ordenadores como tablas organizadas en filas y columnas: hojas de cálculo, bases de datos, entre otras estructuras.

Los autores Catalani y Gutierrez en el 2006, realizaron una descripción de ocho de las bibliotecas disponibles en internet, para resolver problemas de álgebra lineal en computadores de alto rendimiento. Se muestra también una tabla donde se resume las capacidades de 37 bibliotecas de alto rendimiento de álgebra lineal disponibles en internet y de dominio público, esto consolida la importancia que tiene el desarrollo de bibliotecas para la resolución de problemas.

En el artículo *"La experiencia de la UOC con Mathcad"* desarrollado por Álvarez, Molinás, Steegmann, Martínez, y Juan en el año 2002, describen una experiencia docente universitaria basada en la utilización del programa *Mathcad*, mediante el cual, se pretende innovar y mejorar la calidad académica en varias asignaturas de matemática que se ofrecen online en distintas titulaciones de la UOC. En este artículo se puede estudiar un ejemplo para desarrollar un software educativo a partir de un programa computacional de matemática, por otro lado las funciones utilizadas son intrínsecas al programa, por lo que no se puede visualizar el código fuente.

Fundamentación teórica

A continuación se exponen un conjunto de acepciones del álgebra lineal, esenciales para la resolución de problemas relacionados con operaciones algebraicas matriciales. También se definen los diferentes tipos de matrices que se pueden presentar en el álgebra lineal, tanto para matrices rectangulares como para el caso de matrices cuadradas.

Una matriz es un arreglo rectangular de números. Los números del arreglo se conocen como elementos de la matriz. Una matriz A de tamaño m x n es un arreglo rectangular de números dispuestos en m filas y n columnas (Grossman, 2008).

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & & a_{ij} & & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix}$$

Algunos tipos de matrices

A continuación se presenta un conjunto de definiciones de diferentes tipos de matrices (Grossman, 2008), (Lipschutz, 1992).

Matriz fila: Es una matriz que solo tiene una fila, es decir m =1 y por tanto es de orden 1 x n.

$$[a_{11} \quad a_{12} \quad a_{13} \quad \dots \quad a_{1n}]$$

Matriz columna: Es una matriz que solo tiene una columna, es decir, n =1 y por tanto es de orden m x 1.

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{m1} \end{bmatrix}$$

Matriz opuesta: La matriz opuesta de una matriz dada es la que resulta de sustituir cada elemento por su opuesto. La opuesta de A es -A. Lo que quiere decir esto es cambiar los signos de los elementos que forman la matriz, cambiar positivos por negativos.

$$A=\begin{bmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \end{bmatrix},\ -A=\begin{bmatrix} -1 & 2 & -3 \\ 4 & -5 & 6 \end{bmatrix}$$

Matriz nula: Una matriz es nula si todos sus elementos son iguales a cero. En el siguiente ejemplo se muestra la matriz nula de orden 3x2.

$$A=\begin{bmatrix}0 & 0\\0 & 0\\0 & 0\end{bmatrix}$$

La matriz nula, respecto a la adición y multiplicación de matrices, juega un papel similar al número cero respecto a la adición y multiplicación de números reales.

Matriz cuadrada: Es aquella que tiene igual número n de filas que de columnas (n=m). En ese caso se dice que la matriz es de orden n. Por ejemplo, la siguiente matriz es cuadrada de orden 3.

$$A=\begin{bmatrix}0 & 1 & -2\\0 & -3 & 3\\4 & 0.2 & 1\end{bmatrix}$$

Los elementos de la diagonal principal de una matriz cuadrada son aquellos que están situados en la diagonal que va desde la esquina superior izquierda hasta la inferior derecha. En otras palabras, la diagonal principal de una matriz $A= a_{ij}$ está compuesta por los elementos a_{11}, a_{22}, a_{nn}. En el ejemplo anterior la diagonal principal está compuesta por los elementos: $a_{11}= 0$, $a_{22}= -3$, $a_{nn}=1$.

Tipos especiales de matrices cuadradas.

En esta sección se presenta los tipos especiales de matrices. Las primeras matrices son las producidas cuando se aplica el método de eliminación gaussiana a un sistema de ecuaciones lineal (Calderon, 2008; Burden & Faires, 2002; Lipschutz, 1992; Shoichiro, 1992).

Matriz triangular superior y matriz triangular inferior: Una matriz cuadrada se denomina triangular superior (inferior) si todas sus componentes abajo (arriba) de la diagonal principal son cero.

$$A=\begin{bmatrix}1 & 6 & 3\\0 & -2 & 7\\0 & 0 & 5\end{bmatrix}$$

Sí $A=(a_{ij})$ y $B=(b_{ij})$ son matrices triangulares superiores. Entonces:

i. A+B es triangular superior, con diag($a_{11}+b_{11}$, $a_{22}+b_{22}$,…, $a_{nn}+b_{nn}$)
ii. kA es triangular superior, con diag(ka_{11}, ka_{22},…, ka_{nn})
iii. AB es triangular superior, con diag($a_{11}b_{11}$, $a_{22}b_{22}$,…, $a_{nn}b_{nn}$)
iv. A es invertible si y sólo si cada elemento diagonal $a_{ii} \neq 0$.

De manera similar se aplica a las matrices triangulares inferiores.

Matriz diagonal: Una matriz cuadrada D = (d_{ij}) es diagonal si todas sus entradas no diagonales son nulas. Tal matriz se denota frecuentemente por D = diag(d_{11}, d_{22},.... d_{nn}), donde algunos o todos los d_{ii} pueden ser nulos. Por ejemplo.

$$D=\begin{bmatrix}0 & 0 & 0\\ 0 & 6 & 0\\ 0 & 0 & -3\end{bmatrix}$$

La suma, el producto por un escalar y el producto de matrices diagonales son nuevamente diagonales. De este modo, todas las matrices diagonales n x n forman un álgebra de matrices. De hecho, las matrices diagonales forman un álgebra conmutativa, porque dos matrices diagonales n x n cualesquiera conmutan.

Matriz escalar: Es una matriz que tiene todos sus elementos nulos excepto los de la diagonal principal y estos son iguales.

$$A=\begin{bmatrix}5 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 5\end{bmatrix}$$

Matriz unidad: Es una matriz diagonal cuyos elementos de la diagonal son todos 1. A continuación mostramos la matriz unidad de orden 2.

$$I=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

La matriz unidad, respecto a la multiplicación de matrices, juega un papel similar al número 1 respecto a la multiplicación de números reales.

Igualdad de matrices: Dos matrices A y B son iguales si (1) son del mismo tamaño y (2) las componentes correspondientes son iguales.

Matriz simétrica: Una matriz real (cuadrada) es simétrica si $A^t = A$. Equivalentemente, $A = (a_{ij})$ es simétrica si los elementos simétricos (imágenes especulares respecto a la diagonal) son iguales, es decir, si cada $a_{ij} = a_{ji}$.

Matriz antisimétrica: Una matriz real A es antisimétrica si $A^T = -A$. Equivalentemente, una matriz $A = (a_{ij})$ es antisimétrica si cada $a_{ij} = -a_{ji}$. Claramente, los elementos diagonales de una matriz antisimétrica deben ser nulos, ya que $a_{ii} = -a_{ii}$ implica $a_{ii} = 0$. La siguiente matriz es antisimétrica:

$$\begin{bmatrix} 0 & -9 & 3 \\ 9 & 0 & 1 \\ -3 & -1 & 0 \end{bmatrix}$$

Matriz ortogonal: Es aquella cuya traspuesta es igual a su inversa. Es decir, es aquella que multiplicada por su traspuesta da como resultado la matriz unidad. Esto es, A es ortogonal $\Leftrightarrow A \cdot A^T = I \Leftrightarrow A^T = A^{-1}$

La siguiente matriz de funciones es ortogonal:

$$\begin{bmatrix} sen(x) & -\cos(x) \\ \cos(x) & sen(x) \end{bmatrix}$$

Para comprobarlo es suficiente con aplicar la definición y tener en cuenta que

$sen^2 x + \cos^2 x = 1$.

Las matrices ortogonales de orden 2 son de la forma:

$A=\begin{bmatrix} a & b \\ -b & a \end{bmatrix}$ o $A=\begin{bmatrix} a & b \\ b & -a \end{bmatrix}$

donde a y b son números reales tales que $a^2 + b^2 = 1$.

En una matriz ortogonal se cumplen las siguientes afirmaciones:

- La inversa de una matriz ortogonal es una matriz ortogonal.
- El producto de dos matrices ortogonales es una matriz ortogonal.
- El determinante de una matriz ortogonal vale 1 ó -1.

Matriz normal: Una matriz real A es normal si conmuta con su traspuesta, esto es, si $A\ A^t = A^t\ A$. Obviamente, si A es simétrica, ortogonal o antisimétrica, es necesariamente normal. No obstante, éstas no son las únicas matrices normales.

Matriz involutiva: Es una matriz que coincide con su inversa. Esto es,

A es involutiva $\Leftrightarrow A^2 = I$

La siguiente matriz es involutiva:

$$A=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix},\ A^2 = \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

Es evidente que esta matriz también es ortogonal.

Matriz idempotente: Es una matriz igual a su cuadrado. Es decir,

A es idempotente $\Leftrightarrow A^2 = A$

La siguiente matriz es idempotente:

$$A=\begin{bmatrix}1 & 0\\ -1 & 0\end{bmatrix}$$

Matriz nilpotente: Si A es una matriz cuadrada y $A^k = 0$ para algún número natural k, se dice que A es nilpotente. Si k es tal que $A^k - 1 \neq 0$ y $A^k = 0$, se dice que A es nilpotente de orden k. A continuación se tiene una matriz nilpotente de orden 2.

$$A = \begin{bmatrix} 0 & -8 & 0 \\ 0 & 0 & 0 \\ 0 & 5 & 0 \end{bmatrix}, A^2 = \begin{bmatrix} 0 & -8 & 0 \\ 0 & 0 & 0 \\ 0 & 5 & 0 \end{bmatrix}$$

Naturalmente, la matriz A es un divisor de cero.

Matriz tridiagonal: Son aquellas matrices con elementos distintos de 0 en la diagonal y en las posiciones adyacentes (importantes en ecuaciones diferenciales). Esta matriz puede comprimirse a una matriz de 3 columnas y n reglones.

Matriz de diagonal estrictamente dominante: En matemática, y concretamente en álgebra lineal, una matriz es de diagonal estrictamente dominante, cuando lo es por filas o por columnas. Es por fila cuando, para todas las filas, el valor absoluto del elemento de la diagonal de esa fila es estrictamente mayor que la suma de los valores absolutos del resto de elementos de esa fila. Es por columna cuando, para todas las columnas, el valor absoluto del elemento de la diagonal de esa columna es estrictamente mayor que la suma de los valores absolutos del resto de elementos de esa columna.

Formalmente, se dice que la matriz A de n x n es estrictamente diagonal dominante cuando se satisface:

$$|a_{i,i}| = \sum_{j=1, j\neq i}^{n} |a_{i,j}|, \quad \forall \mathrm{i} = \{1, \dots, n\}$$

Matriz positiva definida: Una matriz simétrica A de n x n se llama positiva definida si $x^t Ax > 0$ para todo vector columna n-dimensional $x \neq 0$.

Teorema: Si A es una matriz de n x n positiva definida, entonces A es no singular. Además, la eliminación gaussiana se puede aplicar a cualquier sistema lineal de la forma Ax=b para obtener su solución única sin intercambio de filas o de columnas y los cálculos son estables con respecto al crecimiento de los errores de redondeo.

Operaciones con matrices

En esta sección se realiza una exposición de las diferentes operaciones que se pueden realizar con las matrices, como lo son: la suma, el producto y la traspuesta (Grossman, 2008; Lay, 1999; Lipschutz, 1992).

Suma de matrices: Sean A y B dos matrices de tamaño m x n. Entonces la suma de A y B es la matriz m x n, A + B dada por

$$A = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

Es decir, A + B es la matriz m x n que se obtiene al sumar las componentes correspondientes de A y B.

Es fácil deducir las siguientes propiedades de la adición de matrices de orden m x n:

- Conmutativa: A + B = B + A ∀ A, B ∈ M m x n
- Asociativa: A + (B + C) = (A + B) + C, ∀ A, B, C ∈ M m x n
- Elemento neutro (la matriz nula): ∃ O ∈ M m x n ∀ A ∈ M m x n: A + O = O + A =A
- Elemento opuesto: ∀ A ∈ M m x n ∃ (-A) ∈ M m x n : A + (-A) = (-A) + A = O

Multiplicación de una matriz por un escalar: Si A es una matriz de tamaño m x n y si α es un escalar, entonces la matriz m x n, αA, está dada por

$$\alpha A = \begin{bmatrix} \alpha\, a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha\, a_{21} & \alpha a_{22} & \cdots & \alpha\, a_{2n} \\ \vdots & \vdots & & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha\, a_{mn} \end{bmatrix}$$

Esto es $\alpha A = (\alpha\ a_{ij})$ es la matriz obtenida al multiplicar cada componente de A por α.

Teorema 1

Sean A, B y C tres matrices de m x n y sean α y β dos escalares. Entonces:

- A + 0 = 0 +A = A
- 0A = 0
- A + B=B + A (ley conmutativa para la suma de matrices)
- (A + B) + C= A+(B + C) (ley asociativa para la suma de matrices)
- α(A + B) = αA + αB (ley distributiva para la multiplicación por un escalar)
- 1A = A
- (α + β)A = αA + βA

Nota. El cero en la parte i) del teorema es la matriz cero de m x n. En la parte ii) el cero a la izquierda es un escalar mientras que el cero a la derecha es la matriz cero de m x n.

Multiplicación de matrices: Se denomina matriz producto de la matriz $(a_{ij}) \in$ M m x n por la matriz $B = (b_{ij}) \in$ M m x p a una matriz $C = (c_{ik}) \in$ M n x p cuyos elementos son de la forma

$$C_{ik} = a_{i1}b_{1k} + a_{i2}b_{2k} + \ldots + a_{im}b_{mk} = \sum_{j=1}^{m} a_{ij}b_{jk}$$

Es decir, los elementos que ocupan la posición ik, en la matriz producto, se obtienen sumando los productos que resultan de multiplicar los elementos de la fila i en la primera matriz por los elementos de la columna k de la segunda matriz. A continuación se tiene un ejemplo en el cual se obtiene el elemento c_{23}:

$$A \,.\, B = \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 0 & 4 \end{bmatrix}_{3x2} \begin{bmatrix} -1 & 2 & 5 \\ -2 & 0 & 3 \end{bmatrix}_{2x3} = \begin{bmatrix} -7 & 2 & 14 \\ 0 & 4 & 7 \\ -8 & 0 & 12 \end{bmatrix}_{3x3} = C$$

$$c_{23} = \sum_{j=1}^{2} a_{2j} b_{j3} = a_{21} b_{j3} + a_{22} b_{23} = 2x5 + (-1)x3 = 10 - 3 = 7$$

Dos matrices se pueden multiplicar sólo cuando el número de columna de la primera matriz sea igual al número de filas de la segunda. En ese caso, se dice que las matrices son enlazadas.

Para la multiplicación de matrices no se cumple la propiedad conmutativa. Veamos algunas propiedades de esta operación:

• Asociativa

A.(B.C) = (A.B) .C, ∀ A, B, C: A ∈ M m x n, B ∈ M m x k, C ∈ M k x p

• Elemento neutro (Es la matriz unidad)

∃ I ∈ M n ∀ A ∈ M n : A.I = I.A = A

• Distributiva (mixta)

A.(B+C) = (A.B)+A.C, ∀ A, B, C: A ∈ M m x n, B, C ∈ M m x k

Traspuesta: Sea A una matriz de m x n. Entonces la traspuesta de A, que se escribe A^t, es la matriz de n x m que se obtiene al intercambiar las filas por las columnas de A. De manera breve, se puede escribir $A^t = (a_{ij})$. En otras palabras

$$Si\ A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, entonces\ A^t = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Propiedades

Suponga que A es una matriz de n x m y B es una matriz de m x p. Entonces

i. $(A^t)^t = A$
ii. $(AB)^t = B^tA^t$
iii. $(\lambda A)^t = \lambda A^t$, con $\lambda \in \mathcal{R}$
iv. Si A y B son de n x m, entonces $(A + B)^t = A^t + B^t$
v. Si A es invertible, entonces A^t es invertible y $(A^t)^{-1} = (A^{-1})^t$

Operaciones elementales de matrices

En una matriz, se consideran operaciones elementales por filas a las siguientes:

1. Intercambiar dos filas.
2. Multiplicar una fila por un número real no nulo.
3. Sustituir una fila por la suma de ella misma con el producto de otra por un número real.

Matrices elementales: Se llaman matrices elementales a aquellas matrices cuadradas que resultan de aplicar una operación elemental a la matriz identidad.

Matrices Escalonadas: Una matriz rectangular está en forma escalonada (o en forma escalonada por filas) si tiene las siguientes tres propiedades (Lay, 1999):

1. Todas las filas diferentes de cero están arriba de cualesquier filas con puros ceros.
2. Cada entrada principal de una fila está en una columna a la derecha de la entrada principal de una fila superior.
3. Todas las entradas de una columna que estén debajo de una entrada principal son cero.

Si una matriz en forma escalonada satisface las condiciones adicionales siguientes, entonces está en forma escalonada reducida (o en forma escalonada reducida por filas):

4. La entrada principal de cada fila diferente de cero es 1.
5. Cada 1 principal es la única entrada diferente de cero en su columna.

Factorización

La factorización consiste esencialmente en descomponer la matriz *A* de un sistema de ecuaciones como producto de dos matrices triangulares, siendo los más frecuentes: LU, Doolitle, Crout, Cholesky. La factorización es muy adecuada para aquellas situaciones donde se deben evaluar muchos vectores del lado derecho del sistema de ecuaciones con los mismos coeficientes de la matriz (Chapra & Canale, 2007).

Factorización LU

En esta factorización se descompone la matriz A como producto de dos matrices triangulares A = LU (L es una matriz triangular inferior y U matriz triangular superior). Partiendo del sistema $A \cdot X = B$, se tiene que $L \cdot (U \cdot X) = B$. Se resuelve primero $L \cdot Y = B$ y una vez calculado Y, se resuelve $U \cdot X = Y$, y despejando aquí la X, se obtiene la solución del sistema.

Método de Doolitle

El método de Doolittle obtiene las matrices de la factorización, L y U, fila a fila o columna a columna. Resulta particularmente útil para matrices de grandes dimensiones, o para implementarse en ordenadores con arquitectura en paralelo, el siguiente pseudocódigo adaptado de Kincaid & Cheney, 1994, p.134, representa los pasos para la implementación de este método.

```
Algoritmo Doolitle(a[][], n)
    Para k = 1 hasta n
        Para i = 1 hasta k – 1
            suma=0
            Para p=1 hasta i-1
                suma = suma + l[i][p] * u[p][k]
            Fin Para
            u[i][k] = a[i][k] – suma
        Fin Para
        Para i = k hasta n
            suma=0
            Para p=1 hasta i-1
                suma = suma + l[i][p] * u[p][k]
            Fin Para
            l[i][k] = (a[i][k] – suma) / u[k][k]
        Fin Para
    Fin Para
Fin
```

Método de Crout

En el método de Crout la matriz A es factorizada como A= L.U en donde la matriz L es una matriz triangular inferior y U una matriz triangular superior con diagonal unitaria. Puede advertirse ya aquí una diferencia con el método de Doolittle, en el cual era la matriz L quien poseía diagonal unitaria. El método de Crout es un procedimiento del tipo recursivo, esto significa el desarrollo de un conjunto de pasos sucesivos en donde el trabajo a realizar en cada paso resulta similar o del mismo tipo pero basado en resultados obtenidos en pasos anteriores. Estas "tareas" a realizar en cada paso o "estación" consisten en la descomposición sucesiva de los menores principales de la matriz de coeficientes A. El algoritmo de Crout consta de los siguientes pasos (adaptación de Chapra & Canale, 2007, p292):

```
Algoritmo Crout(a[][], n)
    Para k = 1 hasta n
        Para i = k hasta n
            suma=0
            Para p=1 hasta k-1
                suma = suma + l[i][p]*u[p][k]
            Fin Para
```

```
            l[i][k] = a[i][k] – suma
        Fin Para
        Para i = k+1 hasta n
            suma=0
            Para p=1 hasta k-1
                suma = suma + l[k][p]*u[p][i]
            Fin Para
            u[k][i] = (a[k][i] – suma)/l[k][k]
        Fin Para
    Fin Para
Fin
```

Método de Thomas

El algoritmo de la matriz tridiagonal (TDMA), también conocido como el algoritmo de Thomas, es una forma simplificada de la eliminación de Gauss que se puede utilizar para resolver sistemas de ecuaciones tridiagonal, consiste en tres pasos: descomposición, sustitución hacia adelante y sustitución hacia atrás. El algoritmo de Thomas adaptado de Chapra & Canale, 2007, p.306, está definido por los siguientes pasos:

```
Algoritmo Thomas(a[][], n)
    l[1][1] = a[1][1]
    u[1][2] = a[1][2]  / l[1][1]
    Para i = 2 hasta n - 1
        l[i][i-1] = a[i][i-1]          //(i-ésima fila de L).
        l[i][i] = a[i][i] – l[i][i-1] * u[i-1][i]
        u[i][i+1] =(a[i][i+1])/l[i][i]//((i+1)-ésima columna
                                       //de U).
        l[n][n-1] = a[n][n-1]          //(n-ésima fila de L).
        l[n][n] = a[n][n] – l[n][n-1] * u[n-1][n]
    Fin i
    //Resolver L z = b
    z1 =a[1][n+1]/l[1][1]
    Para i = 2 hasta n
        z[i] = 1 / l[i][i] * (a[i][n+1] – l[i][i-1] * z[i-1])
    Fin i
    //Resolver U x = z
    x[n] = z[n]
    Para i = n – 1 hasta 1
        x[i] = z[i] – u[i][i+1] * x[i+1]
    Fin i
```

```
    SALIDA (x1; x2; : : : ; xn ); PARAR.
Fin
```

Método de Cholesky

Es un procedimiento de factorización desarrollado para aquellos sistemas de ecuaciones en donde la matriz de coeficientes es simétrica y definida positiva. Este tipo de matrices resulta muy usual en la solución de numerosos problemas de contexto matemático y de ingeniería mediante técnicas numéricas como elementos finitos, y de allí la necesidad de incluirlas de modo particular dentro del conjunto de procedimientos de resolución considerados en este capítulo. El método de Cholesky utiliza el preconocimiento de las características mencionadas para realizar una descomposición más eficaz que aquella llevada adelante por el método de Crout. En líneas generales, la secuencia paso a paso considerada por Cholesky es similar a la contemplada por Crout con la significativa diferencia que la matriz A es factorizada como $A=L.L^T$, es decir, ahora solamente los coeficientes de L son incógnitas ya que su traspuesta hace las veces de matriz U en el proceso de descomposición. El algoritmo consta de los siguientes pasos (adaptado del texto Chapra & Canale, 2007, p.309).

Descomposición LL^T – Cholesky

```
Algoritmo Cholesky(a[][], n)
    l[1][1] = (a[1][1])½
    Para k=1 hasta n-1
        l[k+1][1]=a[k+1][1] / l[1][1]
        Para i= 2 hasta k
            s = 0
            Para j=1 hasta i-1
                s = s + l[i][j] * l[k+1][j]
            Fin j
            l[k+1][i]=(a[k+1][i] – s) / l[i][i]
        Fin i
        s=0
        Para i= 1 hasta k
            s = s + (l[k+1][i])2
        Fin i
```

```
        l[k+1][k+1]= (a[k+1][k+1]- s)^½
    Fin k
Fin
```

Sistemas de ecuaciones lineales

El objetivo de esta parte es consultar los diferentes métodos de resolución de sistemas de ecuaciones lineales, que en las carreras de ingeniería se pueden presentar como el resultado del planteamiento de un ejercicio de una red de tuberías para calcular el caudal que circula por las tuberías; una red de circuitos, para calcular las reacciones en estructuras o problemas de sistemas de transporte, entre otras aplicaciones. Atendiendo a la existencia o no de soluciones, los sistemas lineales se clasifican en incompatibles si no tienen solución y compatibles si tienen al menos una solución (Burden & Faires, 2002).

A su vez los sistemas de ecuaciones lineales compatibles se clasifican, en función del número de soluciones, en: determinados si tienen una única solución e indeterminados si tienen más de una, en cuyo caso tendrán infinitas soluciones.

Los sistemas homogéneos tienen siempre, al menos, la solución (0; 0;… ; 0) que recibe el nombre de solución trivial, por ello siempre son compatibles.

Definición de Sistema escalonado

Un sistema de ecuaciones lineales se denomina escalonado (o reducido) si la matriz del sistema verifica que:

(a) Todos los elementos por debajo de los a_{ii} para i = 1; 2; … ; n son nulos.

(b) El primer elemento no nulo de cada fila, llamado pivote, está a la derecha del primer elemento diferente de cero (pivote) de la fila anterior.

(c) Cualquier fila formada únicamente por ceros está bajo todas las filas con elementos diferentes de cero.

Métodos Directos

Los métodos directos de resolución de sistemas lineales de ecuaciones son aquellos que permiten obtener la solución después de un número finito de operaciones aritméticas. Este número de operaciones es, obviamente, función del tamaño de la matriz. Si los ordenadores pudieran almacenar y operar con todas las cifras de los números reales, es decir, si emplearan una aritmética exacta, con los métodos directos se obtendría la solución exacta del sistema en un número finito de pasos. Puesto que los ordenadores tienen una precisión finita, los errores de redondeo se propagan y la solución numérica obtenida siempre difiere de la solución exacta. La cota del error, para una matriz y término independiente dados, se asocia por lo general al número de operaciones de cada método. Se pretende, por lo tanto, obtener métodos con el mínimo número de operaciones posible. Otra particularidad de los métodos directos es que siempre conducen, después de ciertas operaciones, a la resolución de uno o varios sistemas con solución inmediata. Es decir, sistemas donde la matriz es diagonal o triangular. Los métodos para sistemas de resolución inmediata son, de hecho, métodos directos. Una clasificación habitual de estos procedimientos de resolución es aquella que considera que los mismos pertenecen a una de las dos categorías siguientes: métodos de eliminación y métodos de descomposición. En el cuadro siguiente se presenta un esquema con la clasificación de los procedimientos de cálculo más característicos (Bayona Cardenas, 2010).

Clasificación de los métodos directos

- Sistemas con solución inmediata
    - Matriz Diagonal, $A = D$
    - Matriz Triangular Superior, $A=U$
    - Matriz Triangular Inferior, $A=L$
- Métodos de Eliminación
    - Método de Gauss

- Método de Gauss-Jordan

- Métodos de Descomposición
  - Método de Doolittle, A = LU
  - Método de Crout, A = LU
  - Método de Cholesky, $A = L.L^T$
  - Descomposición generalizada, $A=L.D.L^T$
  - Método de Thomas (A Tridiagonal)
- Métodos de ortogonalización

Método de Gauss

Todo sistema lineal Ax = b de n ecuaciones con n incógnitas y $|A| \neq 0$ (o rg A = n) es compatible determinado. Se puede resolver por el método de Gauss:

1. Se considera la matriz ampliada $(A|b)$.
2. Se obtiene una matriz escalonada $(A_r|b_r)$.
3. Se resuelve el sistema equivalente $A_r x = b_r$ por el método de ascenso.

Teorema de Rouché-Frobenius

Si Ax = b es un sistema de m ecuaciones con n incógnitas, entonces:

1. Si $rgA \neq rg\ (A|b)$, el sistema es incompatible.

2. Si $rgA = rg\ (A|b) = n$, el sistema es compatible determinado.

3. Si $rgA = rg\ (A|b) = k < n$, el sistema es compatible indeterminado, y su solución depende de n- k parámetros.

Algoritmo de eliminación gaussiana “simple”.

El Algoritmo de Gauss o de Eliminación gaussiana “simple” consiste en la eliminación de variables hacia adelante y posteriormente se aplica la sustitución

regresiva (adaptado de Chapra & Canale, 2007, p257), este método no realiza el pivoteo de filas, se puede representar mediante los siguientes pasos:

```
Algoritmo eliminacióngaussianasimple(a[][], n)
    //Eliminación hacia adelante
    Para k = 1 hasta n-1
        Para i = k+1 hasta n
            m = a[i][k] / a[k][k]
            Para j = k+1 hasta n
                A[i][j] = a[i][j]- m * a[k][j]
            Fin j
            b[i] = b[i]- m * b[k]
        Fin i
    Fin k
    //Sustitución regresiva
    x[n] = b[n] / a[n][n]
    Para i = n-1 hasta 1, -1
        s = 0
        Para j=i+1 hasta n
            s = s + a[i][j] * x[j]
        Fin j
        x[i] = (b[i] - s) / a[i][i]
    Fin i
Fin
```

Método de Gauss-Jordan

Es una variante del método de Gauss en el cual además de sustraer la fila k multiplicada por $m_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}$ a las filas posteriores, también se sustrae a las anteriores. Es práctica común también en este caso dividir la fila k por el término diagonal a fin de que este quede unitario. El Algoritmo de Gauss-Jordan (adaptado de Shoichiro, 1992, p.185) consta de los siguientes pasos:

```
Algoritmo GaussJordan(a[][], n)
    Para k=1 hasta n
        piv = a[k][k]
        Para j=k hasta n
            a[k][j]= a[k][j] / piv
        Fin j
        b[k] = b[k] / piv
        Para i=1 hasta n
            Si (i != k)
```

```
                m = a[i][k]
                Para j=k+1 hasta n
                    a[i][j]= a[i][j] – m * a[k][j]
                Fin j
                b[i] = b[i]– m * b[k]
            Fin Si
        Fin i
    Fin k
Fin
```

Métodos Iterativos

Un método iterativo es un método que progresivamente va calculando aproximaciones a la solución de un problema. En matemática, en un método iterativo se repite un mismo proceso de mejora sobre una solución aproximada: se espera que lo obtenido sea una solución más aproximada que la inicial. El proceso se repite sobre esta nueva solución hasta que el resultado más reciente satisfaga ciertos requisitos. A diferencia de los métodos directos, en los cuales se debe terminar el proceso para tener la respuesta, en los métodos iterativos se puede suspender el proceso al término de una iteración y se obtiene una aproximación a la solución.

Un método iterativo consta de los siguientes pasos.

1. Inicia con una solución aproximada (Semilla),
2. Ejecuta una serie de cálculos para obtener o construir una mejor aproximación partiendo de la aproximación semilla. La fórmula que permite construir la aproximación usando otra se conoce como ecuación de recurrencia.
3. Se repite el paso anterior pero usando como semilla la aproximación obtenida.

Método de Jacobi

El método Jacobi es el método iterativo para resolver sistemas de ecuaciones lineales más simple y se aplica sólo a sistemas cuadrados, es decir a sistemas con

tantas incógnitas como ecuaciones. El algoritmo (adaptado de Kincaid & Cheney, 1994, p.190) de este método contiene los siguientes pasos:

```
Algoritmo Jacobi(n, a[][], b[], XO[], tol, m)
    k = 1
    Mientras (k <= m)
        Para i = 1 hasta n
            x_i = (b[i] − Σ_{j=1, j≠i}^{n} a[j][j] XO[j]) / a[i][i]
        Fin Para i
        Si (|x[i] - XO[i]| < TOL) entonces
            Imprimir (x[1], x[2] , .. ., x[n])
            PARAR
        Fin Si
        k = k+1
        Para i = 1 hasta n
            XO[i] = x[i]
        Fin Para i
    Fin Mientras
    Imprimir ("Número máximo de iteraciones excedido")
    //Procedimiento completado sin éxito
Fin
```

El Método de Gauss-Seidel

El método de Gauss-Seidel es muy semejante al método de Jacobi. Mientras que en el de Jacobi se utiliza el valor de las incógnitas para determinar una nueva aproximación, en el de Gauss-Seidel se va utilizando los valores de las incógnitas recién calculados en la misma iteración, y no en la siguiente. Por ejemplo, en el método de Jacobi se obtiene en el primer cálculo x_{i+1}, pero este valor de x no se utiliza sino hasta la siguiente iteración. En el método de Gauss-Seidel en lugar de eso se utiliza de x_{i+1} en lugar de xi en forma inmediata para calcular el valor de y_{i+1} de igual manera procede con las siguientes variables; siempre se utilizan las variables recién calculadas. A continuación se muestra un algoritmo del método de Gauss-Seidel (adaptado de Kincaid & Cheney, 1994, p.194).

```
Algoritmo GaussSeidel(n, a[][], b[], XO[], tol, m)
    k = 1
```

```
    Mientras (k <= m)
        Para i = 1 hasta n
            x[i]=(- Σ_{j=1}^{i-1}(a[i][j] x[j]) - Σ_{j=i+1}^{n} a[i][j] XO[j] +b[i])/a_ii
        Fin Para i
        Si (|x[i] - XO[i]| < TOL) entonces
            Imprimir (x[1], x[2] , .. ., x[n])
            PARAR
        Fin Si
        k = k+1
        Para i = 1 hasta n
            XO[i]= x[i]
        Fin Para i
    Fin Mientras
    Imprimir ("Número máximo de iteraciones excedido")
    //Procedimiento completado sin éxito
Fin
```

Operaciones Booleanas

Una matriz booleana es una matriz cuyos elementos son ceros o unos.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Se emplean para representar estructuras discretas (representación de relaciones en programas informáticos, modelos de redes de comunicación y sistemas de transporte) (Calderon Vilca, 2008)

Las operaciones que se pueden realizar entre matrices booleanas son tres: unión, conjunción y producto booleano. Sin embargo, estas operaciones no pueden realizarse sobre dos matrices cualesquiera, sino que deben cumplir ciertos criterios para poder llevarse a cabo. En particular, en el caso de la unión y la conjunción, las matrices que intervienen en la operación deben tener el mismo tamaño, y en el caso del producto booleano, las matrices deben cumplir con las mismas condiciones que para formar el producto de matrices.

Unión / conjunción.

Sean A, B y C matrices booleanas de n x m elementos. Se define $A \vee B=C$ la unión de A y B, por:

$$C[i,j]=\begin{cases}1, \text{ si } A[i,j]=1 \text{ o } B[i,j]=1\\ 0, \text{ si } A[i,j]=B[i,j]=0\end{cases}$$

Intersección / Disyunción.

Sean A, B y C matrices booleanas de n x m elementos. Se define $A \wedge B=C$ la intersección de A y B, por:

$$C[i,j]=\begin{cases}1, \text{ si } A[i,j]= B[i,j]=1\\ 0, \text{ si } A[i,j]=0 \text{ o } B[i,j]=0\end{cases}$$

Matriz complementaria.

La matriz complementaria de A es la matriz cuyos elementos son unos donde A tiene ceros, y ceros donde A tiene unos.

Por ejemplo, la matriz complementaria de $\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$ es $\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix}$

Diferencia simétrica.

La matriz diferencia simétrica de A y B es la matriz booleana cuyo elemento (i, j) es uno cuando el primer elemento es 1 ó 0 y el segundo elemento es el completo; y el elemento (i, j) es cero cuando ambos elementos son ceros o unos.

Por ejemplo, la diferencia simétrica de $\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$ v $\begin{bmatrix}1 & 1 & 1\\ 1 & 0 & 0\end{bmatrix}$ es $\begin{bmatrix}0 & 1 & 0\\ 1 & 1 & 0\end{bmatrix}$

Métodos de Búsqueda

Los computadores se emplean frecuentemente para almacenar grandes volúmenes de datos y tener mucha información lo cual implica dificultades para darle utilidad. Los algoritmos de búsqueda son fundamentales para poder localizar la información relevante en el menor tiempo posible.

La búsqueda lineal o secuencial, consiste en recorrer el arreglo hasta encontrar el elemento buscado (comparando elemento por elemento). Si se encuentra el elemento buscado, se puede mostrar un mensaje "El elemento se ha encontrado"; en caso contrario, visualizar el mensaje "El elemento no se encuentra", para la implementación de este método no es necesario que los elementos estén ordenados. A continuación se muestra el algoritmo de la búsqueda secuencial (adaptación de Joyanes Aguilar, 2008, p376 utilizando un vector), análogamente se puede hacer la búsqueda en matrices.

```
Algoritmo secuencial(v[], n, valor)
    posicion=-1
    i=1
    Mientras (i < n AND v[i] <> valor)
        i=i+1;
    Fin Mientras
    Si (v[i] = valor) Entonces
        posicion= i
    Fin si
Fin
```

El algoritmo de búsqueda secuencial puede ser optimizado si los elementos están ordenados (suponga de forma creciente). En este caso, la búsqueda secuencial desarrollada anteriormente es ineficiente, ya que, si el elemento buscado no se encuentra, se tendría que recorrer todo el vector, cuando se sabe que si se llega a una componente con valor mayor que el elemento buscado, ya no se encontraría dicho valor.

Una primera solución a este nuevo problema sería modificar la condición de salida del ciclo cambiando la condición de igualdad (=) por la condición mayor o igual que (>=), debido a que los elementos se encuentran ordenados de forma creciente:

En la búsqueda binaria, se tiene como restricción que los elementos de la lista deben estar previamente ordenados, con este método se examina primero el valor central; si este es el elemento buscado, entonces la búsqueda ha terminado. En caso contrario se determina si el valor buscado está en la primera o en la segunda mitad del conjunto de valores y, a continuación, se repite este proceso utilizando el elemento central del subconjunto de elementos. Para un conjunto de valores como el siguiente: 0, 1, 2, 3, 8, 14, 18; si se está buscando el número 14, se examina el valor central que es 3, en la cuarta posición. Ya que 14 es mayor que 3, se desprecia el primer subconjunto de valores y se busca en el segundo subconjunto de valores, desde el quinto al séptimo elemento. El número central es 14 el cual coincide con el valor buscado, por lo tanto la búsqueda ha finalizado con éxito.

Este método realiza dos comparaciones, a diferencia que realizando la búsqueda de manera secuencial, se necesitan 6 comparaciones. El procedimiento para buscar un elemento en un conjunto de datos ordenados en forma ascendente sería el siguiente:

1) Si no hay elementos, el programa finaliza.
2) Se calcula el índice del elemento central.
3) Se compara el elemento buscado con el elemento central.
    a) Si el elemento buscado es igual al elemento central se retorna el índice de este, el elemento ha sido encontrado.
    b) Si el elemento buscado es mayor al central, se vuelve al paso 1, pero con el subconjunto a la derecha con índices [central + 1, final].
    c) Si el elemento buscado es menor al central, se vuelve al paso 1, pero con el subconjunto a la izquierda con índices [inicio, central − 1].

Métodos de Ordenamiento

El ordenamiento es una de las aplicaciones computacionales más importante que se utiliza cotidianamente. Su implementación está presente en sistemas bancarios, tiendas comerciales para el manejo de la información de los productos, o también en el manejo de nóminas para administrar la información de los empleados, y en reproductores de música, entre otras aplicaciones. A continuación se presentan algunas definiciones de ordenamiento de datos:

El ordenamiento es el proceso de organizar datos en algún orden o secuencia específica tal como creciente o decreciente para datos numéricos o alfabéticamente para datos caracteres (Joyanes Aguilar, 2008).

El proceso de ordenar consiste en reorganizar un conjunto de objetos en una secuencia específica, con el fin de facilitar la búsqueda de los elementos en el conjunto ya ordenado (Ruiz de Rivillo & Chacón, 1992).

Para poder ordenar una cantidad determinada de números almacenados en un arreglo, existen distintos métodos con distintas características y complejidad, desde el método más simple, como el método burbuja, que son simples iteraciones, hasta el quicksort (método rápido), que al estar optimizado usando recursión, su tiempo de ejecución es menor y es más efectivo.

Los métodos de ordenamiento pueden ser iterativos como: burbuja, inserción, selección y shellsort, estos implementan métodos simples de entender y de programar mediante simples ciclos y sentencias que hacen que el vector pueda ser ordenado. Otro grupo de métodos de ordenamiento que son aún más complejos, requieren de mayor atención y conocimiento para ser entendidos. Son rápidos y efectivos, utilizan generalmente la técnica divide y vencerás, que consiste en dividir un problema grande en varios pequeños para que sea más fácil resolverlos. Entre estos métodos se tiene: ordenamiento por Mezclas (merge) y ordenamiento Rápido (quick).

Ordenamiento Burbuja

El algoritmo de clasificación de intercambio o burbuja se basa en el principio de comparar pares de elementos adyacentes e intercambiarlos entre sí hasta que estén todos ordenados.

Este método consiste principalmente en recorrer n-1 veces los elementos a ordenar, en cada recorrido se compara cada elemento con su sucesor y, si están en orden se mantienen como están, en caso contrario se intercambian entre sí. Por ejemplo para ordenar un conjunto de siete valores en forma ascendente (ver tabla 1), según lo descrito anteriormente en cada recorrido los elementos quedan de la siguiente manera:

Tabla 1. Pasos del método de ordenamiento burbuja.

Estado inicial	**Paso 1**	**Paso 2**	**Paso 3**	**Paso 4**	**Paso 5**	**Paso 6**
3	3	1	1	1	0	0
18	1	2	2	0	1	1
1	2	3	0	2	2	2
2	14	0	3	3	3	3
14	0	8	8	8	8	8
0	8	14	14	14	14	14
8	18	18	18	18	18	18

En el primer paso se observa que primeramente se comprara 3 con 18, como están en orden permanecen en su posición. Seguidamente se compara 18 con 1, como no están en orden, se intercambian. La próxima comparación es 18 con 2 originándose nuevamente un intercambio de posición entre dichos valores, de esta manera se siguen haciendo las comparaciones hasta que finalmente en este primer recorrido, el mayor valor 18 queda al final de los elementos. Este proceso se repite hasta que todos los elementos estén en su respectivo lugar según sea el orden requerido.

Para un vector de n elementos y cuyo índice inicial es 1, se tiene el siguiente algoritmo (adaptación de (Joyanes Aguilar, 2008), p361), dicho método se puede generalizar para el caso de arreglos bidimensionales, realizando los cambios respectivos según se requiera ordenar por fila o por columna.

```
Algoritmo burbuja (v[], n)
    Para i=1 hasta n – 1
        Para j = 1 hasta n - i
            Si (v[j] > v[j+1]) Entonces
                //Se intercambian los valores de la
                //posición j y j+1
                intercambio (v[j] ,v[j+1])
            Fin si
        Fin Para j
    Fin Para i
Fin
```

Ordenamiento por Selección

Este método consiste en seleccionar de un grupo de elementos pertenecientes a una fila o columna de un arreglo, el mayor o el menor valor y colocarlo de último o de primero respectivamente. Luego se busca el segundo elemento mayor o menor y se coloca en la penúltima posición o de segundo, y así sucesivamente hasta que los elementos estén completamente ordenados. Desarrollando este procedimiento paso a paso para un conjunto de n valores en orden ascendente:

1. Seleccionar el elemento menor de todos los elementos que faltan por ordenar.
2. Intercambiar dicho elemento con el primero.
3. Repetir estas operaciones con los n-1 elementos restantes, seleccionando el segundo elemento, continuar con los n-2 elementos restantes hasta que solo quede el mayor valor

Aplicando dicho procedimiento para el mismo ejemplo del método burbuja, se tiene para cada recorrido (ver tabla 2):

Tabla 2. Pasos del método de ordenamiento selección.

Estado inicial	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
3	0	0	0	0	0	0
18	18	1	1	1	1	1
1	1	18	2	2	2	2
2	2	2	18	3	3	3
14	14	14	14	14	8	8
0	3	3	3	18	18	14
8	8	8	8	8	14	18

En el primer paso se busca el menor valor, que es cero y se intercambia con el elemento que está en el primer lugar. Posteriormente se busca el segundo menor valor que es uno y se intercambia con el elemento que está en la segunda posición, y así sucesivamente con el resto de los elementos. El método se puede representar algorítmicamente así (adaptado de Joyanes Aguilar, 2008, p 368):

```
Algoritmo Selección (v[], n)
    Para i = 1 hasta n – 1
        //Función que devuelve la posición del menor valor
        posMen = PosMenor( v, i, n )
        //Función que intercambia los valores de la
        //posición i y posMen
        Intercambio (v[i] ,v[posMen])
    Fin i
Fin
```

Ordenamiento por Inserción

El método de ordenamiento por inserción consiste en ordenar en la primera iteración los dos primeros valores. En la segunda iteración se analiza el tercer valor insertándolo en la posición correcta con respecto a los dos primeros elementos, y así sucesivamente con el resto de los valores hasta que queden los valores totalmente

ordenados. Según lo descrito y aplicándolo al mismo conjunto de valores utilizado como ejemplo en los métodos de ordenamiento anteriores, dichos elementos se ordenan así (ver tabla 3):

Tabla 3. Pasos del método de ordenamiento inserción.

Estado inicial	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
3	3	1	1	1	0	0
18	18	3	2	2	1	1
1	1	18	3	3	2	2
2	2	2	18	14	3	3
14	14	14	14	18	14	8
0	0	0	0	0	18	14
8	8	8	8	8	8	18

En el primer paso se comparan 3 y 18, quedando de la misma manera ya que el orden es ascendente, en el segundo paso, se inserta el 1 en el primer lugar desplazando los valores 18 y 3, de esta forma se tiene ordenado los tres primeros valores; en las siguientes iteraciones se van insertando los valores de la misma manera hasta que finalmente quedan todos ordenados, ver tabla anterior.

Algorítmicamente el método de inserción (adaptado de Joyanes Aguilar, 2008, p363) se puede representar como:

```
Algoritmo inserción (v[], n)
    Para i=1 hasta n-1
        aux = v[i]
        j=i-1
        Mientras (j>=0 AND v[j] > aux)
            v[j+1] = v[j];
            j=j-1
        Fin mientras
        v[j+1] = aux;
    Fin i
Fin
```

Ordenamiento Shell

Es una mejora del método de inserción directa que se utiliza cuando el número de elementos a ordenar es grande. En el método de inserción cada elemento se compara con los elementos contiguos de su izquierda, uno tras otro. Si el elemento a insertar es pequeño, hay que ejecutar muchas comparaciones antes de colocarlo en su lugar definitivamente.

En el método de Shell se modificó los saltos contiguos resultantes de las comparaciones por saltos de mayor tamaño y con eso se conseguía un ordenamiento más rápido. Este método se basa en fijar el tamaño de saltos constantes, pero de más de una posición.

Tomando como ejemplo el mismo conjunto de elementos utilizado para explicar los métodos de ordenamiento anteriores, utilizando el método de Shell ordena los valores de la siguiente manera (ver tabla 4):

Tabla 4. Pasos del método de ordenamiento inserción.

Salto	**Estado inicial**	**Paso 1**	**Paso 2**	**Paso 3**	**Paso 4**	**Paso 5**	**Paso 6**
3	3	2	2	2	0	0	0
3	18	14	14	0	2	2	1
1	1	0	0	3	3	1	2
1	2	3	3	14	1	3	3
1	14	18	18	1	8	8	8
1	0	1	1	8	14	14	14
1	8	8	8	18	18	18	18

A continuación se presenta el algoritmo de Shell, adaptado de Joyanes Aguilar, (2008), p.369.

```
Algoritmo ShellSort (v[], n)
    inc = n;     //incremento
    Hacer
        inc = inc / 2;
```

```
        Para (int k = 0; k < inc; k++)
            i = inc + k
            Mientras (i < n)
                j = i
                Mientras(j- inc >=0 AND v[j]<v[j- inc])
                    intercambio (v[j], v[j- inc])
                    j = j - inc
                Fin mientras
                i = i + inc
            Fin mientras
        Fin para j
    Mientras (inc > 1)
Fin
```

Ordenamiento rápido (QuickSort)

El algoritmo de ordenamiento rápido debido a Hoare, consiste en dividir el vector que se desea ordenar en dos bloques. En el primer bloque se sitúan todos los elementos del vector que son menores que un cierto valor de v que se toma como referencia (valor pivote), mientras que en el segundo bloque se colocan el resto de los elementos, es decir, los que son mayores que el valor pivote. Posteriormente se ordenarían (siguiendo el mismo proceso) cada uno de los bloques, uniéndolos una vez ordenados, para formar la solución.

Evidentemente, la condición de parada del algoritmo se da cuando el bloque que se desea ordenar esté formado por un único elemento, en cuyo caso, obviamente, el bloque ya se encuentra ordenado.

También se puede optar por detener el algoritmo cuando el número de elementos del bloque sea suficientemente pequeño (generalmente con un número aproximado de 15 elementos), y ordenar éste siguiendo alguno de los algoritmos vistos anteriormente (el de inserción directa, por ejemplo).

Aunque cualquier algoritmo de ordenamiento visto anteriormente sea más lento que el Quick Sort, este último pierde gran cantidad de tiempo en clasificar los

elementos en los dos bloques, por lo que, cuando el número de elementos es pequeño, disminuye su eficiencia" utilizar Quick Sort.

Este algoritmo sigue el esquema conocido con el nombre de divide y vencerás que, básicamente, consiste en subdividir el problema en dos iguales pero de menor tamaño para posteriormente combinar las dos soluciones parciales obtenidas para producir la solución global. A continuación se presenta un algoritmo adaptado de Joyanes Aguilar, (2008), p.372.

```
Algoritmo quicksort(v[],inicio, fin)
    //Se define los índices
    i = inicio
    j = fin
    pivote = (L[i] + L[j]) / 2  //Se calcula el pivote
    //Se itera hasta que i sea igual que j
    Mientras (i < j)
        //Se itera mientras que el valor de L[i]
        //sea menor que pivote
        Mientras (L[i] < pivote)
            //Se incrementa el índice
            i=i+1
        Fin mientras
        //Se itera mientras que el valor de L[j] sea mayor
        //que el pivote
        Mientras (L[j] > pivote)
                //Se decrementa el índice
                j=j-1
        Fin mientras
        //Si i es menor o igual que j significa que los
        //índices se han cruzado
        Si (i < = j) entonces
            //Se crea una variable temporal para guardar el
            //valor de L[j]
            x = L[j]
            //Se intercambia los valores de L[j] y L[i]
            L[j] = L[i]
            L[i] = x
            //Se incrementa y decrementa i y j
            //respectivamente
            i+=1
            j-=1
        Fin si
    Fin mientras
```

```
    //Si inicio es menor que j mantenemos la recursividad
    Si (inicio < j)
        L = quicksort(L, inicio, j)
    Fin si
    //Si fin es mayor que i mantenemos la recursividad
    Si (fin > i)
        L = quicksort(L, i, fin)
    Fin si
    //Se devuelve la lista ordenada
Fin
```

Bases Legales

En la actualidad el sistema educativo reclama profesionales de la docencia con una concepción amplia y dinámica del entorno socio-educativo, que puedan contribuir con el desarrollo del país, mediante la colaboración y la participación así como en la orientación de las áreas estratégicas de interés nacional. Por ello se hace necesario innovar los diseños curriculares de las instituciones universitarias permitiendo su adaptación a las necesidades de la nación, lo que se pretende con la incorporación de estrategias instruccionales para el fortalecimiento del futuro docente.

En este sentido se presenta a continuación la fundamentación legal y teórica de la presente propuesta:

Constitución de la República Bolivariana de Venezuela de 1999.

El Preámbulo de (Constitución de la República Bolivariana de Venezuela, 1999), señala: "…con el fin supremo de refundar la República para establecer una sociedad democrática, participativa y protagónica, multiétnica y pluricultural en un Estado de justicia, federal y descentralizado, que consolide los valores de la libertad, la independencia, la paz, la solidaridad, el bien común, la integridad territorial, la convivencia y el imperio de la ley para esta y las futuras generaciones; asegure el

derecho a la vida, al trabajo, a la cultura, a la educación, a la justicia social y a la igualdad sin discriminación ni subordinación alguna…" (p.5)

En dicho artículo se define el ámbito en el cual se deben desarrollar los procesos formativos en el país: una sociedad democrática, participativa y protagónica, multiétnica y pluricultural; basados en esta perspectiva debe anclarse la implementación de un lenguaje de programación, acorde a la tecnología actual en el área de la ingeniería, con la finalidad de que su desarrollo favorezca en el estudiante la formación de valores de libertad, independencia, solidaridad y convivencia. Logrando así la unificación de criterios que garanticen la pertinencia social de nuevas estrategias para la formación integral del estudiante.

En el **artículo 102,** La Constitución establece las características del sistema educativo venezolano, en el cual se establecen claramente los derechos educativos así como, las competencias y atribuciones al Estado Venezolano, a fin de posibilitar el acceso equitativo a una educación de calidad:

> **"…** La educación es un derecho humano y un deber social fundamental, es democrática, gratuita y obligatoria. El Estado la asumirá como función indeclinable y de máximo interés en todos sus niveles y modalidades, y como instrumento del conocimiento científico, humanístico y tecnológico al servicio de la sociedad. La educación es un servicio público y está fundamentada en el respeto a todas las corrientes del pensamiento, con la finalidad de desarrollar el potencial creativo de cada ser humano y el pleno ejercicio de su personalidad en una sociedad democrática basada en la valoración ética del trabajo y en la participación activa, consciente y solidaria en los procesos de transformación social, consustanciados con los valores de la identidad nacional y con una visión latinoamericana y universal. El Estado, con la participación de las familias y la sociedad, promoverá el proceso de educación ciudadana, de acuerdo con los principios contenidos en esta Constitución y en la ley…" (p 92).

En este sentido, se plantea el derecho universal a la educación, y se garantiza el respeto a todas las corrientes del pensamiento, con la finalidad de desarrollar el potencial creativo de cada ser humano y el pleno ejercicio de su personalidad en una

sociedad democrática basada en la valoración ética del trabajo y en la participación activa, fortaleciendo y operacionalizando la perspectiva de los estudiantes, mediante un proceso de formación contextualizado y vinculado a las necesidades socio-educativas reales del país. En relación al cumplimiento de lo señalado en este artículo, la propuesta de la implementación de una herramienta de programación, acorde a la tecnología actual, puede garantizar eficiencia, eficacia y efectividad, si se desarrolla desde una perspectiva crítica y colaborativa.

En el **Artículo 103**, se establece **"**toda persona tiene derecho a una educación integral de calidad, permanente, en igualdad de condiciones y oportunidades, sin más limitaciones que las derivadas de sus aptitudes, vocación y aspiraciones" (p. 31).

En este sentido conviene pensar en mejoras de carácter curricular que favorezcan la formación integral del futuro docente capaz de adquirir nuevos conocimientos y aplicarlos con completa vocación adquiriendo así nuevas destrezas que le garantice su crecimiento y desarrollo como futuro profesional de la docencia.

En el **artículo 108**, se plantea que "…Los centros educativos deben incorporar el conocimiento y aplicación de las nuevas tecnologías y sus innovaciones, según los requisitos que establezca la ley…" (p.97), el cual se complementa con lo estimado en el artículo 110 de la Constitución, en el que "… El Estado reconoce el interés público de la ciencia, la tecnología, el conocimiento, la innovación y sus aplicaciones y los servicios de información necesarios por ser instrumentos fundamentales para el desarrollo económico, social y político del país, así como para la seguridad y soberanía nacional. Para el fomento y desarrollo de esas actividades, el Estado destinará recursos suficientes y creará el sistema nacional de ciencia y tecnología de acuerdo con la ley…" (p 98). Ambos artículos apuntan a la incorporación permanente de innovaciones a los fines de dar mayor calidad y pertinencia social al proceso educativo; siendo el docente el motor principal de cualquier acción educativa, su

propia formación debe estar atravesada por estos criterios y estar en contacto con todos los recursos que el Estado ponga a su alcance para hacerse óptima.

De la Educación Superior

Artículo 27. La educación superior tendrá los siguientes objetivos:

1. Continuar el proceso de formación integral del hombre, formar profesionales y especialistas y promover su actualización y mejoramiento conforme a las necesidades del desarrollo nacional y del progreso científico.
2. Fomentar la investigación de nuevos conocimientos e impulsar el progreso de la ciencia, la tecnología, las letras, las artes y demás manifestaciones creadoras del espíritu en beneficio del bienestar del ser humano, de la sociedad y del desarrollo independiente de la nación.
3. Difundir los conocimientos para elevar el nivel cultural y ponerlos al servicio de la sociedad y del desarrollo integral del hombre.

La (Ley de Universidades, 1970), plantea lo siguiente en sus disposiciones fundamentales:

Artículo 3. Las Universidades deben realizar una función rectora en la educación, la cultura y la ciencia. Para cumplir esta misión, sus actividades se dirigirán a crear, asimilar y difundir el saber mediante la investigación y la enseñanza; a completar la formación integral iniciada en los ciclos educacionales anteriores; y a formar los equipos profesionales y técnicos que necesita la Nación para su desarrollo y progreso.

En este artículo se definen, la misión y el compromiso de las Universidades Nacionales, las cuales deben garantizar la formación de valores trascendentales del hombre así como la colaboración y contribución en la orientación de la vida del país. De igual forma la Universidad de Carabobo, debe formar un profesional eficaz, eficiente, actualizado y comprometido con la sociedad en la cual va a desempeñarse. Por lo tanto debe ofrecer un currículo flexible dirigido hacia el desarrollo integral del estudiante.

CAPÍTULO III

MARCO METODOLÓGICO

En este capítulo se muestran las pautas lógicas y secuenciales pertinentes para el desarrollo y la consecución de los objetivos planteados en el presente trabajo de investigación.

Tipo de Investigación

La presente investigación corresponderá a la modalidad de Proyecto Factible ya que se pretende proponer la creación de una biblioteca de programas, para resolver problemas mediante el uso de arreglos bidimensionales; según (Manual de trabajos de grado de especialización y maestria y tesis doctorales, 2006), el Proyecto Factible "consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El Proyecto debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades.". (p. 21)

El Proyecto Factible comprende las siguientes etapas generales: diagnóstico, planteamiento y fundamentación teórica de la propuesta; procedimiento metodológico, actividades y recursos necesarios para su ejecución; análisis y conclusiones sobre la viabilidad y realización del proyecto; y en caso de su desarrollo, la ejecución de la propuesta y la evaluación tanto del proceso como de sus resultados.

Diseño de la Investigación

La investigación se apoyará en un diseño de campo y documental ya que según Arias (2006), dice que: "la investigación documental es un proceso basado en la búsqueda, recuperación, análisis, crítica e interpretación de datos secundarios, es decir, los obtenidos y registrados por otros investigadores en fuentes documentales: impresas, audiovisuales o electrónicas. Como en toda investigación, el propósito de este diseño es el aporte de nuevos conocimientos.". (p. 27)

Igualmente es importante definir que la investigación de campo, "es aquella que consiste en la recolección de datos directamente de los sujetos investigados, o de la realidad donde ocurren los hechos (datos primarios), sin manipular o controlar variable alguna, es decir, el investigador obtiene la información pero no altera las condiciones existentes. De allí su carácter de investigación no experimental". (Arias, 2006, p.31)

Población

Según Arias (2006), expresa que: "la población, o en término más precisos población objetivo, es un conjunto finito o infinito de elementos con características comunes para los cuales serán extensivas las conclusiones de la investigación. Esta queda delimitada por el problema y por los objetivos del estudio" (p. 81).

La población de la presente investigación estará conformada por los estudiantes del 9no y 10mo semestre de la facultad de ingeniería.

Muestra

Se define la muestra como "un subgrupo de la población. Digamos que es un subconjunto de elementos que pertenecen a ese conjunto definido en sus características al que llamamos población". (Hernández Sampieri, Fernandez Collado, & Baptista Lucio, 2006), p.238. La muestra de la presente investigación estará

constituida por los estudiantes del 9no y 10mo semestre de las diferentes carreras de ingeniería. El muestreo será no probabilístico ya que es un procedimiento de selección en el que se desconoce la probabilidad que tienen los elementos de la población para integrar la muestra. La muestra es de tipo intencionada debido a que el autor de la presente investigación escogerá intencionalmente sus unidades de estudio.

La muestra representa a la población que se le aplicó los instrumentos de recolección de datos, tomando en consideración que según Arocha Correa y López Hernández (2000) señalan que "subconjunto o parte de la población, que se selecciona, se mide y se observa, con el objeto de sacar conclusiones sobre la población". Para obtener el tamaño de la muestra se tomó en cuenta los siguientes aspectos técnicos: Error máximo admisible (E) que definido por Barenson, M y Levine, D (1996) "es el margen que se propone en investigador al realizar un estudio". En este caso se ha establecido un error del ocho por ciento (8%).

El nivel de confianza (Z^2) según Barenson, M y Levine, D (1996) "es el margen de confiabilidad que se propone el momento de una investigación". Se seleccionó un nivel de confianza igual al noventa y cinco (95%).

El valor de (P) según Arocha, C y López, M (2000) expresa que "es la proporción de la población que posee la característica de interés (si no se conoce el valor de esta proporción un valor conservador es P=0,5, debido que es el que produce un mayor tamaño". También expresa que Q es el complemento de P, es decir Q=1-P.

Las fórmulas para determinar el tamaño de la muestra n' es la siguiente:

$$\mathbf{n = \frac{N\,Z^2\,p\,q}{E^2\,(N-1) + Z^2 p\,q}}$$

Donde:

N: población
Z: número de desviación estándar para un nivel de confianza dado (tabla de curva normal)
Probabilidad de éxito y de fracaso. p y q la suma da 1
n: Muestra

En la tabla 5 se muestra la cantidad de graduandos de la Facultad de Ingeniería para el acto de grado en Julio 2012, se estima una cantidad similar de graduandos para el próximo acto.

Tabla 5. Egresados en Julio 2012 por escuela.

Escuela	**Cantidad de Egresados**
Ing. Electricista	96
Ing. Industrial	84
Ing. Mecánico	93
Ing. Químico	105
	N= 378

Fuente: Recuperado de: http://dae.ing.uc.edu.ve/index.php?mod=pagina&node=62

Los datos son:

N : tamaño de la población	378 estudiantes
p : probabilidad de éxito	0,5
q : probabilidad de fracaso	0,5
Z : nivel de confianza 95%	1,96
E : Error máximo admisible 8%	0,08

Se sustituyen los valores para calcular la muestra.

$$n= \frac{378\ (1.96)^2\ 0{,}5 \text{ x } 0{,}5}{(0{,}1)^2\ (378\text{-}1)+\ (1{,}96)^2\ 0{,}5 \text{ x } 0{,}5}$$

$$n= 80 \text{ estudiantes}$$

Selección de los elementos de la muestra

El presente trabajo de investigación utilizará un muestreo probabilístico del tipo aleatorio simple, el cual "es una técnica de muestreo, en el cual cada uno de los miembros de la población tiene alguna probabilidad conocida de ser seleccionado en la muestra" (Arocha Correa & López Hernández, 2000), y con relación al muestreo aleatorio simple los autores lo señalan como "un procedimiento de muestreo que asegura que cada elemento de la población tendrá una probabilidad igual de ser incluido en la muestra". En el caso del presente estudio de una población de 469 estudiantes graduandos de la facultad de ingeniería, a cada estudiante se le aplicará un cuestionario el cual será aplicado en un periodo de una semana, donde se visitarán las escuelas de la facultad para la recolección de los datos.

Técnicas e instrumentos de recolección de datos.

La técnica y el instrumento utilizado para obtener datos o información es la encuesta mediante un cuestionario, la encuesta "como una técnica que pretende obtener información que suministra un grupo o muestra de sujetos acerca de sí mismos, o en relación con un tema en particular" (Arias, 2006).

El instrumento de recolección de datos que será utilizado para obtener, registrar o almacenar información el cual está definido por (Arias, op. cit), como "La modalidad de encuesta que se realiza de forma escrita mediante un instrumento o formato en papel contentivo de una serie de preguntas. Se le denomina cuestionario autoadministrado porque debe ser llenado por el encuestado, sin intervención del encuestador. (p.74).

Con el fin de conocer el punto de vista de los estudiantes, se utilizó la técnica de la encuesta mediante la aplicación de un instrumento tipo test y escala (ver Anexo C); de igual forma se empleó el test a los profesores expertos relacionados con el tema, con el cual se apreciará el punto de vista del docente para la validación del

instrumento, con este fin se le suministro el cuadro de operacionalización de variables (ver Anexo A) y el cuestionario (Ver Anexo D), dicho cuestionario después de su revisión no necesito modificaciones adicionales.

Adicionalmente se realizó una entrevista a expertos en el área de computación e ingeniería con el objetivo de validar el uso de la biblioteca en el área de ingeniería.

Validez y confiabilidad

La confiabilidad se define como "el grado en el que la aplicación repetida de un instrumento de medición al mismo fenómeno genera resultados similares". Del mismo modo se define la validez, "grado en que un instrumento realmente mide la variable que pretende medir" (Hernández Sampieri, Fernandez Collado, & Baptista Lucio, 2006). Para efectos de la investigación la validez se realizará a través de juicios de expertos. La confiabilidad se llevará a cabo, con la aplicación del método estadístico, coeficiente de alfa de Cronbach, el cual requiere una sola administración del instrumento de medición y producen valores que oscilan entre 0 y 1. Su ventaja reside en que no es necesario dividir en dos mitades a los ítems del instrumento de medición y se calcula el coeficiente. La manera de calcular este coeficiente se explica en el siguiente.

$$rtt = \frac{k}{(k-1)\left[\frac{1-\sum Si^2}{St^2}\right]}$$

Donde:

rtt : coeficiente de confiabilidad de la prueba o cuestionario.

k: número de ítems del instrumento.

st^2: Varianza total del instrumento.

Σsi^2: Sumatoria de las varianzas de los ítems.

Cuanto menor sea la variabilidad de respuesta, es decir, que haya homogeneidad en las respuestas dentro de cada ítem, mayor será el Alfa de Cronbach. Por ejemplo en la tabla 6, se realiza el cálculo del alfa de Cronbach de un cuestionario, mediante la herramienta IBM SPSS Statics 21, obteniéndose como resultado un valor de 0.76 el cual es aceptable. Para establecer el nivel satisfactorio de confiabilidad suele recurrirse a valores mayor a 0.70 ó 0.8 (dependiendo de la fuente), que presuponen un escaso condicionamiento del porcentaje de correlaciones por el error aleatorio de la medida para escalas de uso amplio, por lo que se considera una fuerte confiabilidad.

Tabla 6. Ejemplo de las respuestas a los ítems del cuestionario.

	Items											
Alumnos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**13**	**Suma**
1	1	4	1	1	2	5	1	3	1	…	1	26
2	2	3	3	3	3	3	3	3	3	…	3	38
3	1	4	4	4	4	4	3	3	3	…	4	45
4	3	4	2	2	3	3	1	3	3	…	4	37
5	2	1	1	4	4	5	2	3	3	…	4	38
6	2	4	4	3	2	2	3	2	3	…	4	39
7	1	2	2	2	4	4	1	3	3	…	2	31
8	3	3	3	3	5	5	1	2	3	…	3	39
9	2	4	4	4	4	5	4	3	4	…	4	52
10	3	2	2	2	2	4	2	4	4	…	2	26
…	…	…	…	…	…	…	…	…	…	…	…	…
80	2	2	2	2	3	3	3	2	4	…	2	37
Σxi	173	255	213	239	255	282	177	190	221	…	243	3011
Σxi²	443	925	687	829	919	1128	477	534	697	…	855	10235
Si²	0,87	1,42	1,52	1,46	1,34	1,70	1,08	1,05	1,09	…	1,48	16,35

Fuente: Propia.

St2	54,08
Alpha	**0,76**

Fases Metodológicas.

Para dar cumplimiento al conjunto de objetivos planteados como marco de este trabajo de investigación se procederá a ejecutar las siguientes actividades:

Fase 1

Realizar un análisis al estado del arte mediante un proceso de revisión bibliográfica y levantamiento de información, referente a las necesidades esenciales de implementación de matrices en la resolución de problemas, en el área de computación y de ingeniería. Esto es:

- Investigar cuales bibliotecas existen en la actualidad, cuál es su alcance de aplicación y el lenguaje implementado.
- Determinar cuáles son los procedimientos básicos usuales en la resolución de problemas que necesitan la implementación de arreglos bidimensionales en el álgebra matricial y computación.
- Evaluación del cuestionario diseñado, por expertos en el área de matemática y computación, para el diagnóstico de la necesidad de la biblioteca de métodos; y su posterior aplicación a estudiantes de los últimos semestres de la carrera a excepción de ingeniería civil.

Fase 2

Diseñar mediante un modelo orientado a objetos, la estructura, clases y métodos de la biblioteca para la creación, identificación, cálculos y operaciones matriciales que se emplean en el área de computación y de ingeniería. Estos métodos son desarrollados tomando en cuenta la diversidad de tipos de datos con los cuales se puede trabajar en el lenguaje y adicionalmente verificar la posibilidad de implementar parámetros que puedan resolver casos específicos no tratados en otras bibliotecas. Los métodos a desarrollar en forma general incluyen operaciones elementales,

aritméticas, de clasificación, generación, booleanas, resolución de sistemas de ecuaciones, búsqueda y ordenamiento.

La selección de los métodos se hace en base a los resultados obtenidos del cuestionario y de la opinión a expertos mediante entrevista personal.

Fase 3

Crear la biblioteca para el manejo de arreglos diseñada, utilizando el paradigma de programación orientada a objetos y las técnicas de manejo de matrices apropiadas.

- El desarrollo del programa será en lenguaje Java, mediante el editor NetBeans 7.2.1.
- Cada archivo de clase contiene un conjunto de métodos, cada uno de ellos permite realizar actividades específicas para la resolución del problema.
- El archivo de clase Matriz, contiene los constructores necesarios para la implementación de la biblioteca.
- Cada uno de los métodos tiene el archivo de prueba correspondiente en el cual se invocan los métodos correspondientes para la comprobación de sus resultados.

Fase 4

Realizar pruebas de funcionamiento de la biblioteca, basadas en su integración y uso en una aplicación para el área de ingeniería. Dichas pruebas se realizan mediante la ejecución desde el método principal realizando las actividades usuales en la resolución de un problema del área de ingeniería o computación.

Se seleccionó dos ejercicios resueltos de un texto y se resolvió implementando la biblioteca de clases, comparando finalmente los resultados obtenidos con los del libro para validar los resultados.

Recursos Institucionales

La institución involucrada en esta investigación es: La Facultad de Ingeniería de la Universidad de Carabobo; mediante la colaboración de su personal docente y en particular los Departamento de Computación y Matemática, las bibliotecas digitales de las diferentes universidades nacionales e internacionales.

Recursos Humanos

El recurso humano disponible está integrado por Profesores del departamento de computación y profesores del departamento de matemáticas de la facultad de ingeniería de la Universidad de Carabobo.

Recursos Económicos y Financieros

El proyecto tendrá un financiamiento propio, en cuanto a los recursos materiales a utilizar; un computador de escritorio que tenga instalados los programas Microsoft Office 2010 y el editor de java NetBeans IDE 7.2.1.

CAPÍTULO IV

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En este capítulo se presenta el análisis e interpretación de los resultados obtenidos luego del procesamiento de la información recolectada mediante los instrumentos diseñados para tal fin. Adicionalmente se realiza el estudio de factibilidad respectivo de soporte para la implementación de la propuesta y se explica en que consiste la biblioteca de clases en java para resolver problemas de computación y algebra matricial, se realiza una representación esquemática de los diferentes métodos desarrollados, así como también su respectivo encabezado y una descripción del uso del método y la definición de cada una de los argumentos.

Análisis y discusión de resultados

La aplicación del cuestionario a estudiantes graduandos de las diferentes escuelas, representan el soporte fundamental para las propuestas realizadas en este proyecto. El grupo de estudiantes bajo medición están cursando sus últimas asignaturas de la carrera o están por entregar su proyecto de grado. El cuestionario se validó mediante la participación de varios expertos, docentes en el área de computación y matemática, en el anexo E se muestra el formato de validación utilizado por el docente para evaluar la claridad y congruencia del cuestionario, con el fin de dar soporte al diagnóstico y a la necesidad de desarrollar la biblioteca contemplado en los dos primeros objetivos de este trabajo. Análogamente se realizó entrevistas a docentes para sustentar la necesidad del desarrollo de la biblioteca para resolver problemas en el área de matemática y computación mediante el uso de matrices. En la tabla 7 se muestra el resultado obtenido para cada uno de los ítems del cuestionario.

Tabla 7. Cantidad de respuestas para cada ítem.

Cantidad de respuestas por pregunta													
Item	1	2	3	4	5	6	7	8	9	10	11	12	13
Totalmente en desacuerdo	24	7	17	9	6	9	5	28	16	12	3	21	9
En desacuerdo	24	18	21	21	19	10	9	15	7	30	15	19	20
Ni de acuerdo, ni en desacuerdo	27	19	20	22	19	11	17	29	37	23	15	29	20
De acuerdo	5	25	16	18	26	30	36	8	20	14	35	11	21
Totalmente de acuerdo	0	11	6	10	10	20	13	0	0	1	12	0	10
Porcentaje (%) de respuestas por cada pregunta													
Totalmente en desacuerdo	30	9	21	11	8	11	6	35	20	15	4	26	11
En desacuerdo	30	23	26	26	24	13	11	19	9	38	19	24	25
Ni de acuerdo, ni en desacuerdo	34	24	25	28	24	14	21	36	46	29	19	36	25
De acuerdo	6	31	20	23	33	38	45	10	25	18	44	14	26
Totalmente de acuerdo	0	14	8	13	13	25	16	0	0	1	15	0	13

Fuente: Propia

A continuación se analiza cada una de las respuestas al cuestionario aplicado a los estudiantes. En lo que respecta al diagnóstico del uso de las matrices en aplicaciones prácticas de la ingeniería, se tiene de los ítems del 1 al 13 lo siguiente:

1. Prefiero escribir totalmente el código, aunque ya este creado. Con respecto a esta pregunta se tiene que alrededor de un 60% de los alumnos prefiere escribir el código, desde el punto de vista de aprendizaje es la recomendación que el profesor le da a los alumnos para que desarrollen la habilidad de creación de programas, el resto de los alumnos aprovechan en lo posible código fuente previamente escrito, sin embargo el estudiante debe verificar que dicho código realice las actividades pertinentes.
2. Es conveniente disponer de subprogramas para crear una aplicación. Más del 45% de estudiantes opinan que si es conveniente, la implementación de subprogramas es una técnica de programación muy utilizada ya que permite dividir el programa en grupos de actividades que resuelven una parte del problema, resolviendo de esta forma el problema en su totalidad, invocando cada uno de los subprogramas.

Para el estudiante los subprogramas revisten mayor importancia cuando deben realizar un programa para un proyecto o el trabajo de grado, ya que la cantidad de código a desarrollar es mayor, los subprogramas les facilita la edición de código, en particular si logran conseguir una biblioteca de código relacionado con su trabajo.

3. Es conveniente descargar o copiar el código necesario de cualquier fuente en internet para crear una aplicación para un proyecto en el computador. Más de un 47% de encuestados manifiestan no estar de acuerdo con realizar la descarga o copia de código en internet, en el caso de asignaciones o trabajos en clase es frecuente que las actividades a realizar no sean de uso común, por lo que escasamente podrá conseguir dicho código en internet, también se puede presentar el caso de que el código tenga errores o este incompleto.
4. En las asignaturas de la carrera, se requiere del uso del computador para resolver problemas. Un 37% de los encuestados opinan que no se requiere utilizar el computador para la resolución de problemas, en la actividad de clase los ejercicios están planificados para resolver manualmente los ejercicios, en las asignaturas no se cuenta con software educativos que permitan realizar operaciones más complejas de casos prácticos e inclusive simulaciones para que el estudiante tenga una visión más general del tema de estudio.
5. Las aplicaciones actuales se deben desarrollar para trabajar en cualquier sistema operativo y para móviles. Más de un 45% opinan estar de acuerdo con que las aplicaciones se deben desarrollarse para multiplataforma y dispositivos móviles, precisamente una de las herramientas más utilizadas en esta área es el lenguaje de programación Java, la facilidad de poder utilizar aplicaciones en diferentes dispositivos como PC, celulares o tabletas permite al estudiante tener una mayor disponibilidad o capacidad de acceso a la tecnología.
6. Es útil disponer de algún subprograma para el ordenamiento de datos. Un 62% de los encuestados consideran que es útil tener un subprograma de ordenamiento de

datos, ya que es una actividad muy utilizada en el área de computación, estadística, búsqueda de datos, entre otras aplicaciones.

7. Es necesario disponer de un subprograma para la resolución de sistemas de ecuaciones lineales. Más del 61% de estudiantes considera que es necesario ya que en asignaturas del ciclo básico, en electivas y obligatorias de la carrera requieren de la resolución de sistemas de ecuaciones lineales.
8. En ocasiones es necesario buscar algún valor en un arreglo. Un 53 están en desacuerdo, la actividad de búsqueda de datos es común en asignaturas de computación pero en el resto de las asignaturas tienen un uso muy limitado.
9. En el manejo de matrices es importante poder reconocer que tipo de matriz se está utilizando. Un 29% está en desacuerdo, esto en gran parte se debe a que en las asignaturas en las que se requiere resolver un sistema de ecuaciones los ejercicios son de un tipo tal que la solución se puede dar por cualquiera de los métodos de resolución o siempre aplican un método en particular.
10. Un sistema de ecuaciones lineales, se puede resolver mediante la implementación de cualquier método de resolución. Un 52% están en desacuerdo ya que los diferentes métodos de resolución tienen ventajas dependiendo del tipo de sistema de ecuaciones a resolver.
11. Mi trabajo de grado, requerirá la resolución de problemas mediante álgebra matricial o búsquedas u ordenamientos. Más de un 58% de encuestados consideran que necesitaran implementar arreglos para aplicarlo en alguna actividad referida a su trabajo de grado.
12. La facultad de ingeniería de la Universidad de Carabobo, cuenta con aplicaciones que permiten resolver problemas de álgebra matricial, búsqueda, ordenamiento y resolución de sistemas de ecuaciones lineales. Más del 49% de los encuestados consideran que no se cuenta con herramientas computacionales de aplicación en el manejo de arreglos bien sea para resolver problemas ni como software educativo ni simuladores.

13. Existe software comercial que realizan operaciones matriciales, ordenamiento, búsqueda y resolución de sistemas de ecuaciones lineales. El 38% conoce de la existencia de aplicaciones comerciales que realizan actividades de cálculo mediante el uso de matrices.

Estos resultados certifican la necesidad del desarrollo de la biblioteca, asimismo la opinión de expertos investigadores en el área, de la Facultad de Ingeniería.

El profesor D. Rey Lago, Director del Instituto de Matemática y Cálculo Aplicado (IMYCA) considera que:

> El empleo de matrices para la resolución de problemas es de gran importancia en mis áreas específicas de investigación (entrevista personal, Febrero 15, 2013).
>
> 1. Cómputo de alto rendimiento y procesamiento paralelo: una gran parte de los grandes problemas de ciencia e ingeniería que requieren cómputo intensivo, se fundamentan en el álgebra de matrices densas y dispersas. Las matrices de estos problemas se utilizan principalmente para representar sistemas de ecuaciones, grillas espaciales y vectores de estado.
>
> 2. Ingeniería eléctrica, área de sistemas digitales: las matrices se utilizan para representar las salidas de los circuitos digitales para todas las combinaciones de entradas (tabla de la verdad), representación interna de algoritmos tabulares de optimización, para representar el comportamiento de una máquina de estado (tabla de transición de estados), etc.
>
> Efectivamente, esta biblioteca se puede utilizar como base para el desarrollo de futuras aplicaciones mediante programas de procesamiento paralelo.

El profesor C. Jiménez (entrevista personal, Febrero 15, 2013) Director de la Escuela de Ingeniería Eléctrica, indica que:

> En el pensum de la escuela que dirige hay asignaturas donde es primordial el uso de matrices y que el estudiante debe dominar:
>
> 1) La generación de modelos matemáticos de los sistemas eléctricos utilizando matrices.

2) La clasificación y operaciones elementales de las matrices.
3) Conocer las diferentes operaciones matriciales.

Entre las asignaturas donde se evidencia esta necesidad tenemos:

- Circuitos Eléctrico I
- Circuitos Eléctrico I
- Mediciones Eléctricas
- Teoría de control III
- Simulación
- Electrónica III
- Sistemas de Potencia I
- Sistemas de Potencia I
- Electrotecnia (Asignatura cursada por Las escuelas de Industrial, Mecánica y Química)
- Procesamiento Digital de Señales (Pregrado y Maestría)

Otras Aéreas de Aplicación:

1) Procesamiento de Imágenes, en donde todas las señales, imágenes se modelan a través de matrices para realizar transformación y análisis. (Doctorado)
2) Mecánica Racional
3) Estructuras
4) Operaciones unitarias
5) Programación Lineal (Maestría)
6) Análisis Espectral (Doctorado)

El profesor E. Flores (entrevista personal, Febrero 15, 2013) coordinador de la Maestría de Matemática y Computación, y Director de Extensión menciona lo siguiente:

> El cálculo matricial tiene una amplia aplicación en el área de ingeniería y otras ciencias como la economía, biología, etc. La biblioteca puede servir de apoyo a investigaciones del pregrado y postgrado, la creación de software educativo para la enseñanza y el aprendizaje del álgebra lineal entre otras asignaturas que impliquen el uso del cálculo matricial. Las aplicaciones alcanzan áreas como: grafos, análisis de esfuerzo, computación gráfica, redes de fluidos entre otras aplicaciones.

J. Jiménez profesor del Departamento de Matemática (entrevista personal, Febrero 15, 2013) menciona que:

> La biblioteca de métodos suaviza la curva de aprendizaje del investigador. Esto es posible teniendo un código confiable que cubra las operaciones usuales para el desarrollo de un trabajo de investigación, esto evita tener que invertir tiempo en aprender el uso o manejo de diferentes bibliotecas o paquetes informáticos.
>
> Los métodos incluidos en la biblioteca tienen amplia aplicación en el pregrado y también el postgrado como por ejemplo en el tratamiento de señales, programación paralela, entre otras aplicaciones.

El profesor W. Sanz (entrevista personal, Febrero 15, 2013) jefe del departamento de sistemas y automática de la escuela de ingeniería eléctrica comenta que:

> En mi campo de trabajo particular (Robótica) el empleo de matrices se utiliza para la aplicación de transformaciones que permiten modelar la cinemática y dinámica de robots. También, simplifican en gran medida el manejo conjunto de ecuaciones para la determinación de parámetros correspondientes al movimiento y elementos de los robots.
>
> Resulta por consiguiente, incuestionable la utilidad del uso de las matrices en la resolución de problemas.
>
> Considero que varios de los mostrados son de importancia significativa en el pregrado de ingeniería. A continuación listo los que me parecen más relevantes:
>
> - Operaciones: Matricial: Todas las indicadas.
> - Operaciones: Escalar: Determinante, Traza.
> - Métodos para resolver ecuaciones lineales: ambos métodos de eliminación y el de sustitución.
> - Métodos de Búsqueda: Fila/columna: secuencial
> - Métodos de ordenamiento: burbuja

El profesor T. Rojas (entrevista personal, Febrero 15, 2013) comenta lo siguiente:

> Soy Ingeniero Electricista con grado de Maestría en la carrera y mi disciplina es la "Teoría de Control". No obstante, puedo dar la opinión de que efectivamente las matrices son elementos básicos y muy importantes en la representación de los sistemas eléctricos y físicos en general (Teoría de Control Moderna), siendo entonces necesarias las herramientas de

cálculo algebraico para su solución. Por ejemplo, en el caso de redes de transmisión los parámetros del sistema quedan asociados a matrices complejas y para sistemas de control multivariables, también su representación paramétrica se hace a través de matrices.

LA BIBLIOTECA DE MÉTODOS

La biblioteca está conformada por un conjunto de clases (ver figura 1) que permiten realizar la creación de matrices de un tipo determinado, identificarla o clasificarla según sus valores, realizar modificaciones en la matriz como producto de operaciones elementales en la matriz, determinar el resultado matricial o escalar de las operaciones con matrices, buscar y ordenar valores en las matrices, factorizar matrices y resolver sistemas de ecuaciones lineales, en la siguiente figura se muestra una representación de las clases implementadas en la biblioteca y su dependencia entre ellas.

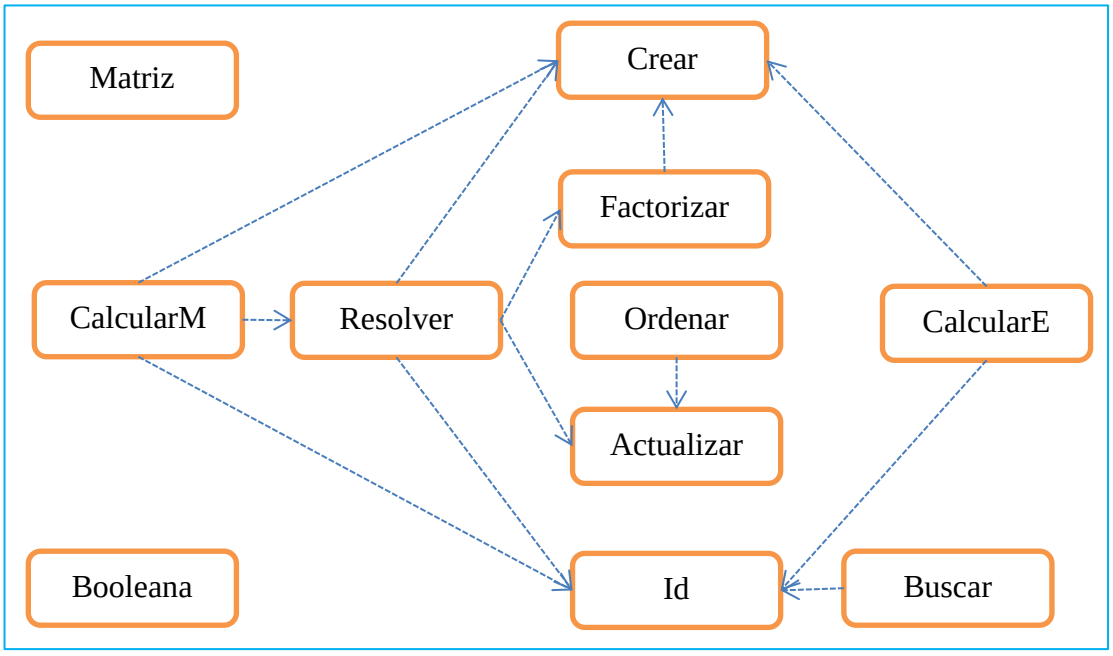


Figura 1. Representación esquemática de las clases y su dependencia.

La clase Matriz, constructores y miembros

La clase matriz está conformada por 8 métodos, en la misma se declara las variables miembros de la clase de nombre *mat* como un arreglo bidimensional de tipo objeto con *rows* cantidad de filas y *cols* cantidad de columnas, ambas inicializadas en cero, a continuación se detallan cada uno de los constructores implementados en la clase *Matriz*.

Los métodos `public Matriz(Object[][] a)` y `public Matriz(Matriz a)` permiten crear el objeto *Matriz*, a partir de un arreglo bidimensional de tipo *Object* o del mismo tipo del objeto *Matriz*.

Para crear un objeto *Matriz* de una dimensión determinada, se pueden utilizar el siguiente constructor `public Matriz(int cf, int cc)` sí se requiere un arreglo rectangular donde *cf* es la cantidad de filas y *cc* es la cantidad de columnas o cuadrado mediante el constructor `public Matriz(int n)`, en el otro constructor *n* representa la dimensión de una matriz cuadrada.

En caso de crear un objeto *Matriz* de un determinado tipo de dato entero o real, se implementa el constructor `public Matriz(int cf, int cc, boolean e1)` en el cual se define su tamaño con la cantidad de filas *cf* y cantidad de columnas *cc* si es rectangular y si es cuadrada con el constructor `public Matriz(int n, boolean e1)`, se define el tamaño mediante el parámetro entero *n*, en ambos constructores mediante el parámetro *e1* de tipo booleano se define la inicialización de la matriz con valores de tipo entero si es verdadero o real si el valor es falso.

Finalmente se puede crear una matriz mediante el constructor `public Matriz(Matriz a, boolean e1)` utilizando como parámetro una matriz conocida, se utiliza sus dimensiones para crear la matriz y el parámetro *e1* de tipo booleano define la inicialización de la matriz con valores de tipo entero si es verdadero o real si el valor es falso.

Creación de arreglos bidimensionales

La creación de matrices (conformada por 30 métodos) es fundamental para la resolución de problemas mediante el uso del álgebra matricial, en la biblioteca se desarrolló un conjunto de métodos para la creación de matrices necesarias para la resolución de problemas, dichas matrices son creadas:

- Provenientes de un archivo de texto, lo cual en la práctica, se obtiene proveniente de datos de equipos, de páginas web, de otros programas, los cuales almacenan los datos mediante archivos de texto.
- Por teclado, si necesariamente se debe digitalizar los datos directamente por este dispositivo.
- De valores aleatorios, en el caso de experimentos o simulaciones, este tipo de matrices consisten en definir el menor y el mayor valor (rango) de los valores a generar para el llenado del arreglo.
- Duplicado de una matriz.
- Llena de ceros o unos para iniciar en dicho valor todas las celdas del arreglo.
- Una submatriz a partir de una matriz dada.
- Matriz opuesta de una matriz determinada.
- Una matriz aumentada, obtenida a partir de una matriz y un vector.

La matriz cuadrada representa un caso particular, de las cuales se tiene en consideración los siguientes métodos:

- Diagonal, en la cual se asigna un valor determinado, contenido en un vector, a cada celda de la diagonal principal.
- Escalar, permite crear la matriz con un valor determinado en todas las celdas de la diagonal principal.

- Identidad, es un caso particular en la cual la matriz se encuentra llena de unos en la diagonal principal.
- Hilbert, la cual es una matriz cuadrada cuyos campos constituyen una fracción de la unidad.

En el anexo F.1 se muestra de forma general el diseño de la clase *Crear*, la cual permite definir la forma o tamaño de la matriz al igual que sus valores iniciales. Para ello se dispone de un conjunto de métodos con sus respectivos parámetros que permiten crear la matriz según sea necesario.

Rectangular

`static Matriz aleatoria(int cf,int cc, boolean e1, Object li, Object ls)`, devuelve un arreglo de dimensiones *cc* y *cf* con números aleatorios desde un valor inferior *li* hasta un valor superior *ls* todos de tipo entero o real dependiendo del valor de la variable *e1*.

`static Matriz aumentada(Matriz a, Matriz x, boolean e1)`, devuelve la matriz aumentada como resultado de unir la matriz *a* con la matriz *x*, ambas con la misma cantidad de filas.

`static int cantFilas(String ruta) throws FileNotFoundException, IOException`, mediante este método se determina la cantidad de filas contenidas en el archivo definido por el parámetro *ruta*.

`static Matriz ceros(int cf,int cc,Boolean e1)`, devuelve una matriz nula de un tamaño definido por los parámetros *cf* y *cc*, y con valores de un tipo de dato entero o real.

`static Matriz delArchivo(String ruta)`, devuelve una matriz leída desde un archivo especificado en el parámetro *ruta*, este método verifica la

cantidad de filas de la matriz y verifica el tipo de dato de la matriz posteriormente crea la matriz y le asigna los valores leídos línea por línea del archivo.

`static Matriz delTeclado(int cf, int cc)`, recibe como parámetros la cantidad de filas y de columnas, se crea el arreglo y se habilita la lectura por teclado para llenar la matriz, recorriéndola por filas.

`static private double[][] dimensiones(String archivo)`, es un método alternativo para conocer la dimensión de la matriz, es decir la cantidad de filas y de columnas.

`static Matriz duplicada (Matriz a)`, devuelve una matriz identica en tamaño y contenido a la matriz recibida como parámetro.

static Matriz eliminar(Matriz a,char foc, int k), devuelve una matriz con la fila o columna eliminada según el parámetro *foc*, en la posición *k*.

`static Matriz girarH(Matriz a)`, devuelve la matriz *a* girada horizontalmente.

`static Matriz girarV(Matriz a)`, devuelve la matriz *a* girada verticalmente.

`static Matriz insertarCMatriz(Matriz a, Object b[], int k)` y `static Matriz insertarFMatriz(Matriz a, Object b[], int k)`, permiten crear una matriz en la cual se inserta en la columna o fila k el vector *b* en la matriz *a*.

`static Matriz insertarCMatriz(Matriz a, Matriz b, int columnainicio)` y `static Matriz insertarFMatriz(Matriz a, Matriz b, int filainicio)`, permiten crear una matriz en la cual se unen

las matrices *a* y *b* a partir del parámetro *filainicio* si se desea insertar a partir de una fila o *columnainicio* para insertar la matriz b a partir de una columna especifica.

`static public double[][] lee(String archivo)`, permite leer desde el archivo definido en el parámetro *archivo* para crear una matriz cuyo tamaño se obtiene mediante el método *dimensiones* y posteriormente almacenar en el arreglo todos los valores leídos desde el archivo.

`static Matriz matrizVector (Matriz a, int cf, int cc)`, devuelve una matriz en fila con una cantidad de elementos igual al producto de la cantidad de filas de la matriz *a* por su cantidad de columnas.

`static Matriz opuesta(Matriz a)`, devuelve la matriz opuesta de la matriz *a*.

`static Matriz subMatriz (Matriz a,int fi,int ci, int cf, int cc)`, devuelve una matriz producto de la extracción de una submatriz con inicio en la posición de la celda fila *fi* y columna *ci*, la submatriz tiene una tamaño igual al definido en los parámetros *cf* y *cc* que significan cantidad de fila y cantidad de columnas respectivamente.

`static Matriz unos(int cf, int cc, boolean e1)`, devuelve una matriz de tamaño *cf* x *cc*, llena de unos, de tipo entero si el parámetro *e1* es verdadero y de tipo real si es falso.

`static Matriz vectorMatriz (Matriz a, int cf, int cc)`, devuelve una matriz con un tamaño definido por los parámetros *cf* y *cc* que representan la cantidad de filas y de columnas del arreglo a crear.

Cuadrada

`static Matriz diagonal(int n,Object k, Object v[], boolean e1)`, este método devuelve un arreglo de tamaño n x n, inicialmente crea un arreglo correspondiente a una matriz nula y posteriormente recorre la diagonal principal del arreglo y le asigna los valores contenidos en el vector *v*.

`static Matriz escalar(int n,Object k, boolean e1)`, este método crea inicialmente una matriz llena de ceros y posteriormente recorre la diagonal principal para asignarle el valor *k* a todos sus elementos, sí el parámetro *e1* recibe el valor true entonces los valores asignados son de tipo entero en caso contrario son de tipo real.

`static Matriz identidad(int n, boolean e1)`, devuelve una matriz cuadrada de tamaño n x n con unos en la diagonal principal y el resto de las celdas con ceros, todos ellos de tipo entero sí *e1* es verdadero y real sí es falso. Este método para realizar esta operación invoca al método escalar.

`static Matriz hilbert(int n, boolean e1)`, devuelve una matriz cuyos elementos son iguala a una fracción correspondiente a 1/(i+j+1).

Prueba de la biblioteca para la creación de las matrices.

En la figura 2 se muestra una ejecución de la prueba de los métodos para la creación de diferentes tipos de matrices como por ejemplo: matriz identidad, escalar, duplicar, Hilbert, entre otras.

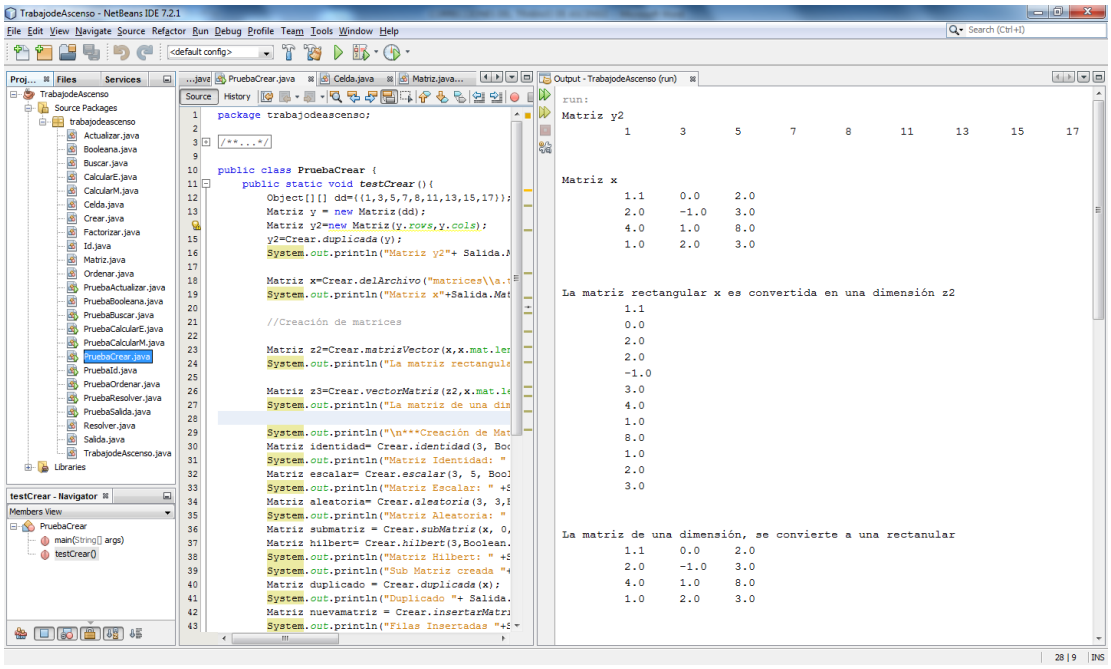


Figura 2. Ejecución de la clase Crear para clasificar una matriz.

Identificación de arreglos bidimensionales

Durante el proceso de cálculo con operaciones algebraicas y en computación es necesario realizar comparaciones para identificar la matriz que se está procesando, con el fin de tomar una decisión para implementar un método o técnica determinada. Tal es el caso de la resolución de un sistema de ecuaciones lineal cuya matriz correspondiente es una matriz triangular superior, el método de resolución más adecuado es el de sustitución regresiva, o en el área de computación para realizar una búsqueda binaria se debe tener los datos previamente ordenados.

Esta clase conformada por métodos que permiten mejorar la eficiencia de programas, también se puede implementar para el desarrollo de software educativo en el proceso de identificación de los diferentes tipos de matrices que se emplean en el área de matemática y de computación, en el anexo F.2 se muestran todos los métodos

desarrollados de la clase *Id* para la identificación de las matrices. A continuación se muestra la sintaxis de cada método y la explicación de su funcionamiento.

`static boolean tipoEntero(Matriz a)` y `static boolean tipoEntero(Object z)`, ambos métodos identifican si la matriz recibida como parámetro es de tipo entero, esto lo hace mediante la verificación de todos los elementos, si al menos una celda contiene un valor de un tipo de dato diferente entonces devuelve falso, si todos los valores contenidos son enteros, devuelve verdadero.

`static int tipodeDato(Object z)`, devuelve un valor entero correspondiente al tipo de dato del valor *z* recibido como parámetro.

`static int tdMatrizArchivo(String ruta) throws FileNotFoundException, IOException`, devuelve el tipo de dato asociado al arreglo contenido en un archivo de texto ubicado en la dirección especificada en el parámetro *ruta*.

`static boolean cuadrada(Matriz a)`, devuelve verdadero si la matriz *a* es cuadrada y falso en el caso de que sea rectangular.

`static boolean igualTamaño(Matriz a, Matriz b)`, realiza la comparación de las matrices *a* y *b*, verificando que tengan la misma cantidad de filas y de columnas, en caso afirmativo devuelve verdadero, si alguna de las dos dimensiones es diferente devuelve falso.

`static boolean antisimetrica(Matriz a)`, verifica que los elementos a_{ij} de la matriz sean igual a los elementos $-a_{ji}$ de la misma matriz.

`static boolean booleana(Matriz a)`, verifica si todos los elementos de la matriz son ceros y unos.

`static boolean diagonal(Matriz a)`, verifica que los elementos de la diagonal principal de la matriz a sean diferente a cero y el resto igual que cero.

`static boolean diagonalmenteDominante(Matriz a)`, verifica si la matriz es diagonalmente dominante.

`static boolean escalar(Matriz a)`, verifica si los elementos de la diagonal principal son iguales y diferente a cero, devuelve true si la matriz es escalar de lo contrario devuelve false.

`static boolean escalonada(Matriz a)`, verifica si el arreglo es una matriz escalonada.

`static boolean idempotente(Matriz a)`, verifica si el producto de la matriz a por sí misma es igual a la matriz a.

`static boolean identidad(Matriz a)`, verifica si los elementos de la diagonal principal son todos igual a uno.

`static boolean iguales(Matriz a, Matriz b)`, compara celda a celda los valores de dos matrices, devuelve true si todos los valores son iguales y false si al menos un valor es diferente. Las matrices a comparar deben ser de igual tamaño.

`static boolean involutiva(Matriz a)`, verifica si el producto de la matriz a por sí misma es igual a la matriz identidad.

`static boolean nula(Matriz a)`, verifica si todos los componentes del arreglo son igual a cero.

`static boolean opuesta(Matriz a, Matriz b)`, método para determinar si los elementos del arreglo bij son igual a los elementos -aij.

`static boolean ordenada(Matriz a, int foc, int pos, int inicio, int fin, int orden)`, en este método se verifica sí la matriz está ordenada por fila o columna, de igual manera se especifica cual es el índice de la fila o de la columna, además se define entre cuales índices de la otra dimensión y el orden a verificar si es ascendente o descendente.

`static boolean ortogonal(Matriz a)`, verifica si el producto de la matriz a por su traspuesta es igual a la matriz identidad.

`static boolean simetrica(Matriz a)`, verifica que los elementos a_{ij} de la matriz sean igual a los elementos a_{ji} de la misma matriz.

`static boolean subMatriz (Matriz a,Matriz b, int x, int y)`, verifica si el arreglo b está contenido en el arreglo a, a partir de la celda ubicada en la posición x, y.

`static boolean triangularInf(Matriz a)`, verifica si todos los elementos que están por encima de la diagonal principal son igual a cero.

`static boolean triangularSup(Matriz a)`, verifica si todos los elementos que están por debajo de la diagonal principal son igual a cero.

`static boolean tridiagonal(Matriz a)`, verifica si la matriz es tridiagonal.

Prueba de la biblioteca para la identificación o clasificación de las matrices.

En la ejecución de la clase de identificación de matrices, se creó una matriz de nombre *z*, en dicha matriz se revisa sus componentes para identificar el tipo de matriz, luego se verifica si la matriz esta ordenada y adicionalmente se comprueba si está escalonada o si es una matriz diagonal.

Seguidamente se crean dos matrices identificadas como *x*, *y*; a las cuales se les asignan valores a partir de la información contenida en un archivo de texto, finalmente se realizan otro conjunto de comprobaciones para clasificar la matriz o también para realizar comparaciones entre ellas. Un ejemplo de la salida en consola de la ejecución del método *Id*, se presenta en la figura 3.

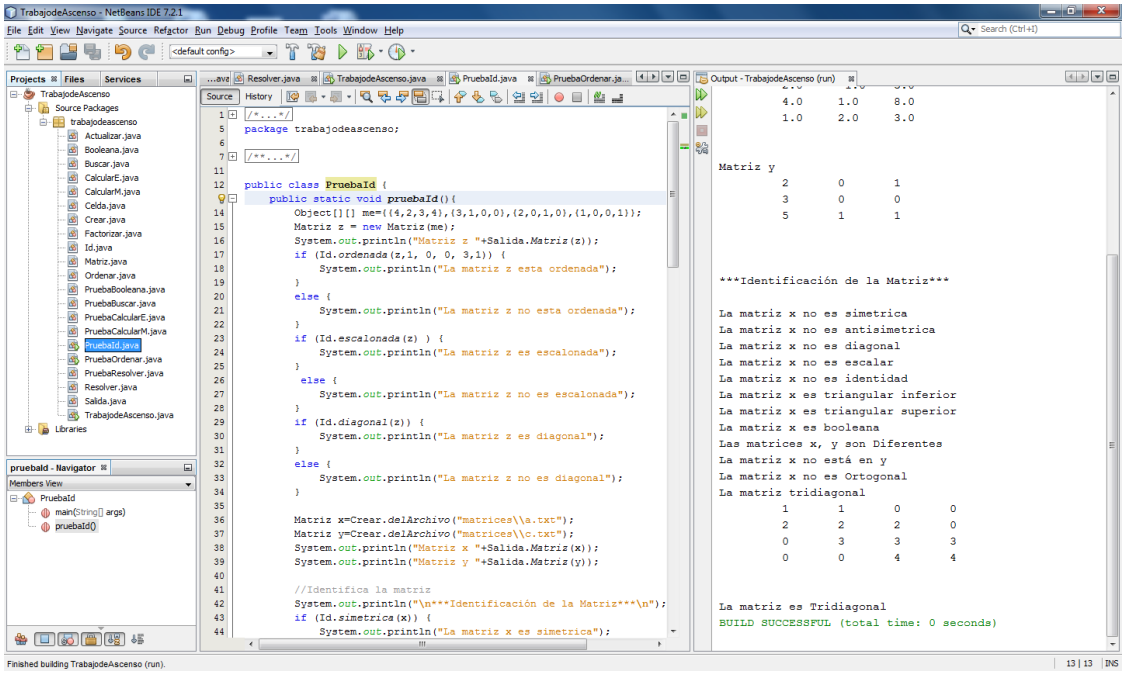


Figura 3. Ejecución de la clase Id para clasificar una matriz.

Operaciones booleanas

Una matriz booleana es una matriz cuyos elementos son ceros o unos. Se emplean para representar estructuras discretas (representación de relaciones en programas informáticos, modelos de redes de comunicación y sistemas de transporte), ver anexo F.3.

`static Matriz complementaria(Matriz a)`, retorna la matriz complementaria de la matriz a.

`static Matriz diferenciaSimetrica(Matriz a, Matriz b)`, devuelve una matriz producto de la diferencia simétrica de las matrices *a* y *b*.

`static Matriz interseccion(Matriz a, Matriz b)`, devuelve una matriz con los elementos producto de la intersección de las matrices *a* y *b*, verifica elemento por elemento si son iguales asigna uno a la matriz a crear sino asigna a el valor cero.

`static Matriz union(Matriz a, Matriz b)`, devuelve una matriz con los elementos producto de la unión de las matrices *a* y *b*, verifica elemento por elemento los elementos de las matriz *a* y *b*, si alguno de los dos elementos es igual a uno, se le asigna uno a la matriz a crear sino le asigna el valor cero.

Prueba de la biblioteca para el trabajo con matrices booleanas.

En la ejecución de los métodos de operaciones boolenas, se crearon dos matrices a las cuales se le realizó la operación de unión e intersección, en la figura 4 se observa los resultados obtenidos por consola.

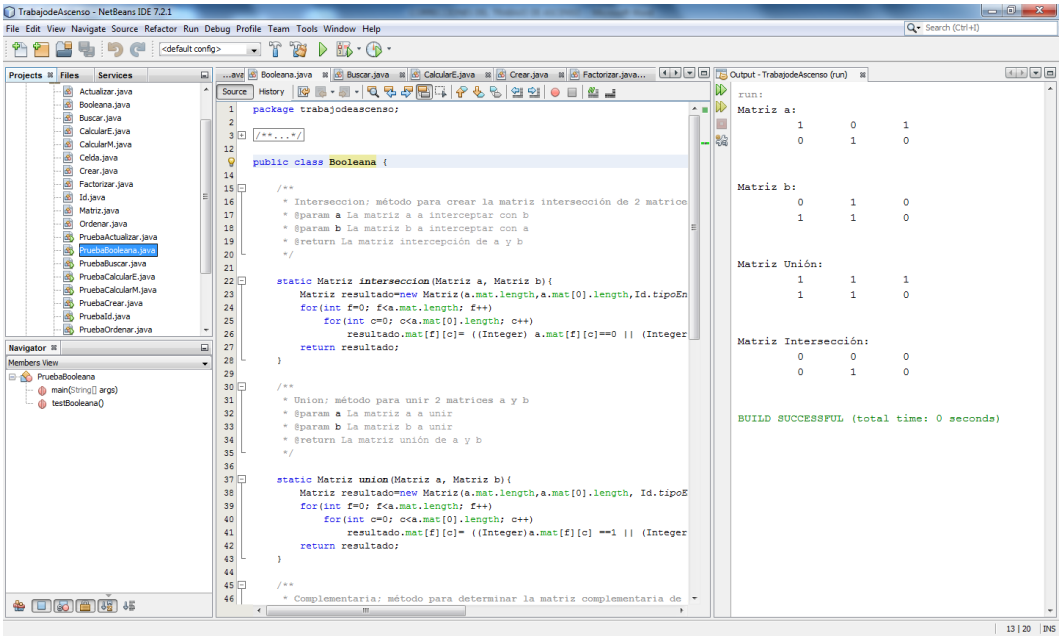


Figura 4. Ejecución de las operaciones booleanas.

Operaciones elementales para actualizar arreglos bidimensionales

Las operaciones elementales en matrices incluidas en la clase Actualizar (ver anexo F.4) permiten modificarla de tal manera que se pueda resolver el sistema de ecuaciones lineal. En el proceso de resolución de sistemas de ecuaciones lineales mediante el empleo de matrices, se requiere de operaciones de actualización como: intercambio de filas, multiplicación de una fila por un escalar, sumar dos filas y remplazar una de ellas.

`static Matriz multiplicarFilaporK(Matriz a, Object k, int fila)`, devuelve la matriz *a* con los valores de una fila multiplicados por un valor *k*. En este método el parámetro *fila* define la fila de la matriz a multiplicar y el parámetro *k* representa el escalar por el cual se multiplicará la fila.

`static Matriz sumarFilas(Matriz a, int k, int l)`, este método permite sumar filas, se define *k* y *l* que son las filas a sumar en la matriz *a*, el método devuelve la matriz *a* cuya fila *l* se le suma la fila *k*.

`static Matriz sumarFilas(Matriz a, int k, double c, int l)`, este método suma a la fila *l*, el resultado de multiplicar la fila *k* por el valor *c*.

`static void intercambiar(Matriz a, char foc, int k, int l)`, En el método Intercambiar se puede tanto intercambiar filas como columnas, este último en el caso de organizar datos mostrados por columnas, para ello se dispone del parámetro *foc* el cual es de tipo char cuyos valores son *f* o *c*, los enteros *k* y *l* representan las filas o columnas a intercambiar, a continuación se tiene el encabezado del método, el resultado es la matriz a modificada.

`static void intercambiar(Matriz a, int f1, int c1, int f2, int c2)`, también se cuenta con un método adicional que permite

intercambiar dos celdas de una matriz, los parámetros *f1*, *c1* y *f2*, *c2* representan los índices de las dos celdas a intercambiar.

Prueba de la biblioteca para las operaciones de actualización.

En la prueba de la clase *Actualizar*, se realizó en una matriz las operaciones de producto de un escalar por una fila, intercambio de filas y de columnas, suma de filas y añadir a una fila el resultado de multiplicar otra fila por un escalar, en la figura 5 se muestra el conjunto de actividades desarrolladas en la prueba.

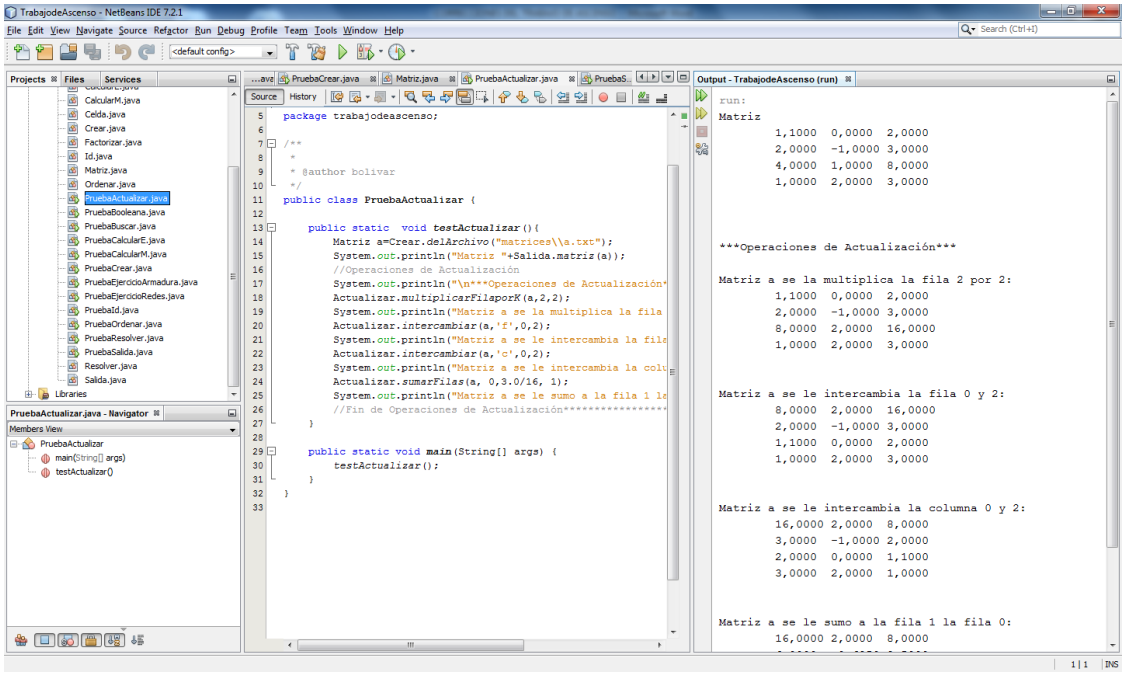


Figura 5. Ejecución de la clase actualizar.

Cálculos escalares y matriciales

A continuación se presentan dos clases que contienen los métodos que permiten realizar operaciones de cálculo básico que dan como resultado una matriz (contiene 6 métodos, ver anexo F.5) o un escalar (contiene 20 métodos, ver anexo F.6).

Matricial

Los siguientes métodos conforman la clase CalcularM, devuelven un arreglo como resultado de la operación correspondiente.

`static Matriz inversa(Matriz d)`, devuelve a matriz inversa de la matriz *d*, en caso de no tener inversa, devuelve el valor *null*.

`static Matriz potencia(Matriz a, Object k)`, eleva el contenido de la matriz a un valor k.

`static Matriz producto(Matriz a, Matriz b)`, devuelve una arreglo como resultado del producto matricial de las matrices *a* y *b*.

`static Matriz producto(Matriz a,Object k)`, devuelve una matriz como resultado del producto del valor k por los elementos del arreglo *a*.

`static Matriz suma(Matriz a, Object k)`, suma un valor *k* a cada elemento del arreglo a.

`static Matriz suma(Matriz a, Matriz b)`, este método devuelve un arreglo como resultado de la suma matricial de la matriz *a* y la matriz *b*.

`static Matriz traspuesta(Matriz a)`, devuelve un arreglo que corresponde a la traspuesta de la matriz *a*, cada elemento a_{ij} de la matriz *a* es el elemento a_{ji} de la matriz traspuesta.

Prueba de la biblioteca para el cálculo matricial.

En la figura 6 se visualiza la ejecución de los métodos de la clase de cálculos matriciales entre los cuales se tiene suma de matrices, producto de matrices y producto de un escalar por una matriz, entre otras operaciones.

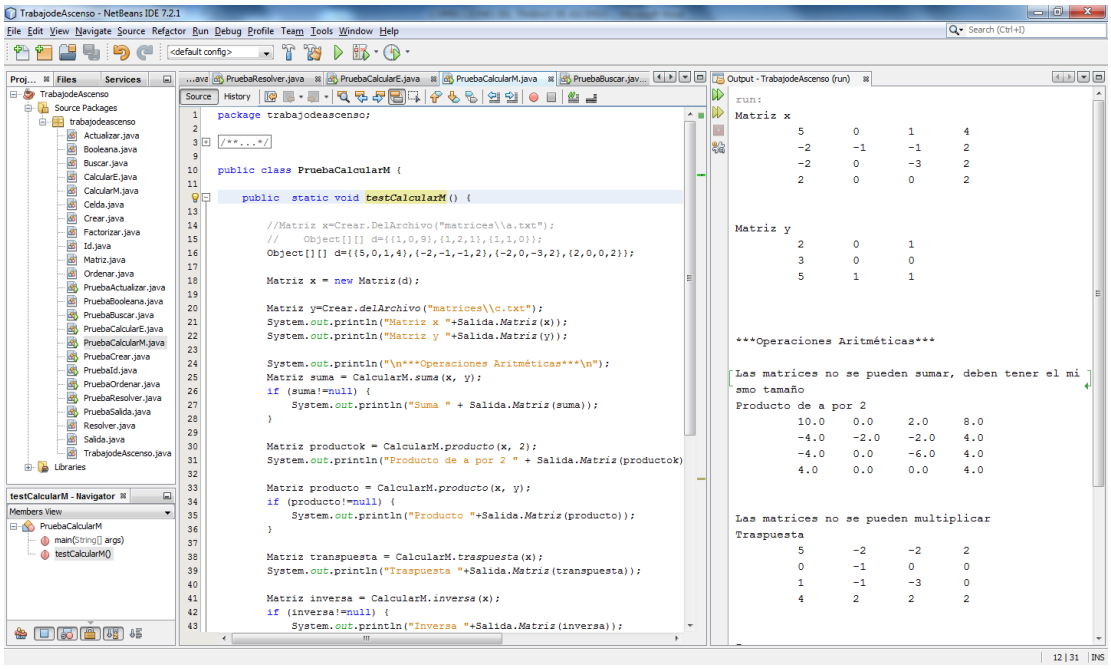


Figura 6. Ejecución de la clase Prueba CalcularM.

Escalar

Los siguientes métodos están contenidos en la clase CalcularE, devuelver un valor escalar como resultado de sus operaciones.

`static int cantidad(Matriz a, int foc, int pos, Object k)`, devuelve la cantidad de valores igual a un valor k que se encuentran en la fila o columna según "foc" y en el índice "pos".

`static int cantidad(Matriz a, Object k)`, devuelve la cantiodad de valores igual a k que se encuentran en la matriz a.

`static double determinante(Matriz a)`, devuelve el valor del determinante de la matriz *a*.

`static Object maximo(Matriz a)`, devuelve el máximo valor contenido en la matriz *a*.

`static Object maximo(Matriz a, int foc, int pos)`, devuelve la posición del máximo valor de una fila o columna determinada.

`static Object minimo(Matriz a)`, devuelve el mínimo valor contenido en la matriz *a*.

`static Object minimo(Matriz a, int foc, int pos)`, devuelve la posición del mínimo valor de una fila o columna determinada.

`static double norm(Matriz a)`, devuelve la norma de la matriz *a*.

`static Object suma(Matriz a, int foc, int pos)`, devuelve la suma de los elementos de una fila o columna de la matriz a definido por el parámetro *foc*, y para un índice determinado por el parámetro *pos*.

`static Object sumaCuadrado(Matriz a)`, permite determinar la suma al cuadrado de los componentes del arreglo

`static Celda posMaximo(Matriz a)`, devuelve la fila y columna de la celda en la matriz *a* que contiene el primer máximo valor.

`static int posMaximo(Matriz a, int foc, int pos)`, devuelve la posición del máximo valor de una fila o columna determinada.

`static Celda posMinimo(Matriz a)`, devuelve la fila y columna de la celda en la matriz *a* que contiene el primer mínimo valor.

`static int posMinimo(Matriz a, int foc, int pos)`, devuelve la posición del mínimo valor de una fila o columna determinada.

`static double promedio (Matriz a)`, devuelve el promedio de los valores contenidos en la matriz *a*.

`static double promedio (Matriz a, int foc, int pos)`, devuelve el promedio de los elementos contenidos en una fila o columna definida por el parámetro *foc* y con un índice igual al valor contenido en el parámetro *pos*.

`static Object suma (int k, int l, Matriz a)`, devuelve la suma de los elementos que están alrededor de la celda ubicada en la posición *k, l*.

`static Object suma (Matriz a)`, devuelve la suma de todos los elementos de la matriz a, si todos los elementos de la matriz *a* son de tipo entero entonces el resultado será entero, de lo contrario será de tipo real.

`static double traza (Matriz a)`, devuelve el valor de la traza de la matriz.

Prueba de la biblioteca para el cálculo escalar en matrices.

En la figura 7 se muestra la ejecución de los métodos de la clase para el cálculo de un escalar a partir de una matriz, se realizan las operaciones para calcular suma de valores en una fila, suma de valores en una columna, promedio, los valores máximo y mínimo al igual que la posición de dichos valores, la cantidad de valores igual a un valor determinado bien sea por fila o por columna, etc.

Factorización de matrices

La factorización LU de una matriz es una factorización que resume el proceso de eliminación gaussiana aplicado a la matriz y que es conveniente en términos del número total de operaciones de punto flotante cuando se desea calcular la inversa de una matriz o cuando se resolverá una serie de sistemas de ecuaciones con una misma matriz de coeficientes, en el anexo F.7 se muestra los métodos de factorización desarrollados en la clase *Factorizar*.

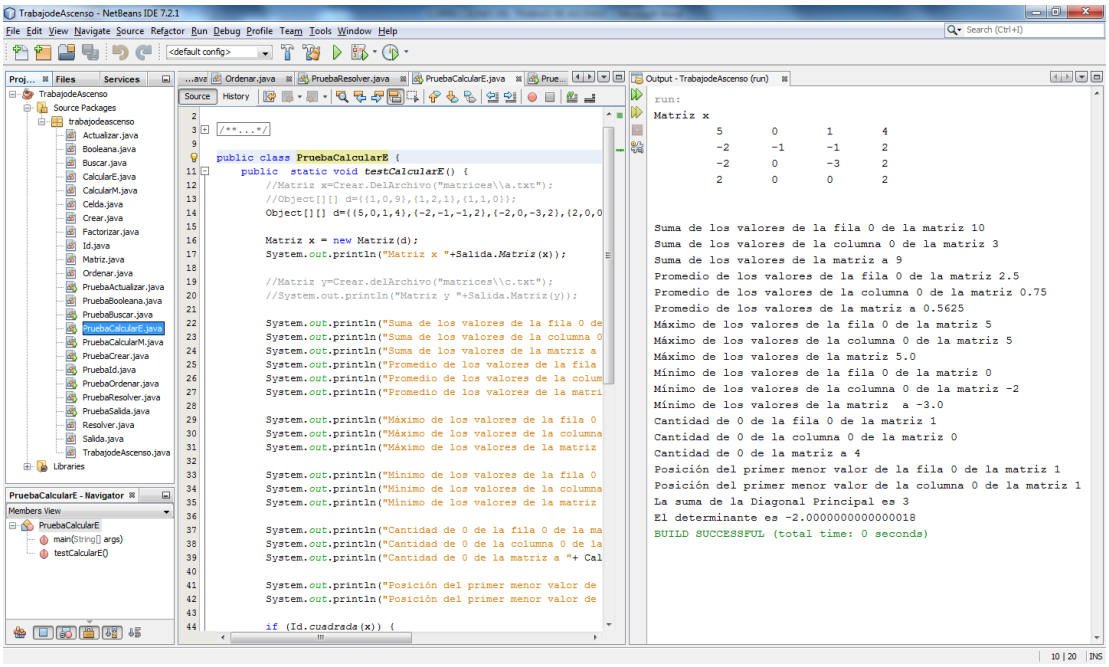


Figura 7. Ejecución de la clase CalcularE.

A continuación se muestra la sintaxis de los métodos de descomposición de la biblioteca, los cuales reciben la matriz a y devuelven como resultado las matrices l y u.

```
static void cholesky(Matriz a, Matriz l, Matriz u)

static void crout(Matriz a, Matriz l, Matriz u)

static void doolitle(Matriz a, Matriz l, Matriz u)

static void lu(Matriz a1, Matriz l, Matriz u)

static void thomas (Matriz a1,Matriz l, Matriz u)
```

El parámetro a es la matriz a factorizar, la matriz l es una matriz triangular inferior y la matriz u es una matriz triangular superior, ambas son devueltas por el método.

Métodos de resolución de sistemas de ecuaciones lineales

Resolver un sistema de ecuaciones es determinar todas sus soluciones. Obviamente, un sistema se puede resolver cuando es compatible, es decir, lo primero que se debe hacer es averiguar su compatibilidad (teorema de Rouché-Fróbenius).

Para resolver un sistema de ecuaciones lineal se tiene que realizar transformaciones en las ecuaciones hasta que todas las incógnitas queden despejadas. Estas transformaciones convierten el sistema inicial en otro sistema (con aspecto distinto y más fácil de resolver) que tienen las mismas soluciones (sistemas equivalentes).

Generalmente las transformaciones más habituales son:

- Intercambiar dos ecuaciones entre sí.
- Suprimir una ecuación que tenga todos sus elementos nulos.
- Suprimir una ecuación que sea proporcional a otra.
- Suprimir una ecuación que sea combinación lineal de otra/s
- Multiplicar o dividir una ecuación por un número distinto de cero.
- Sustituir una ecuación i de este modo : $E_i = E_i + a\ E_j$

Los métodos de resolución de sistemas de ecuaciones lineales directos obtienen la solución del sistema de ecuaciones en un número finito de operaciones.

`static Matriz gauss(Matriz a)`, el método de Gauss aplicado en un conjunto de n ecuaciones con n incógnitas se reduce a un sistema triangular equivalente (un sistema equivalente es un sistema que tiene iguales valores de la solución), que a su vez se resuelve fácilmente por sustitución inversa.

`static Matriz sustitucionRegresiva(Matriz a)` y `static Matriz sustitucionProgresiva(Matriz a)`, el método de sustitución

regresiva permite calcular los componentes de la solución, despejando la última incógnita de la última ecuación, con la cual se sustituye en la penúltima ecuación; después se despeja de dicha ecuación la penúltima incógnita y se repite el proceso hasta calcular el valor de la primera incógnita, a continuación se muestra la sintaxis de los métodos de sustitución regresiva y progresiva, ambas devuelven un arreglo de tipo columna con la solución del sistema.

Los siguientes métodos invocan a los métodos de factorización y devuelve la solución del sistema definido por la matriz a.

```
static Matriz cholesky (Matriz a1)
```

```
static Matriz crout (Matriz a1)
```, U tiene diagonal de 1.

```
static Matriz doolitle (Matriz a1)
```, L tiene diagonal de 1.

```
static Matriz lu (Matriz a)
```

```
static Matriz thomas (Matriz a1)
```

Los métodos iterativos son menos sensibles a los errores de redondeo y esto se aprecia en sistemas de orden elevado donde los errores de redondeo de los métodos directos son considerables.

```
static void jacobi(Matriz a, Matriz b, Matriz x0,
Matriz x, float max)
```

Prueba de la biblioteca para la resolución de sistemas de ecuaciones lineales.

La prueba de verificación de los resultados utilizando los métodos de resolución de sistemas de ecuaciones lineales, se realizó mediante la creación de una matriz y su posterior utilización en un método de resolución, en la figura 8 se muestra los resultados obtenidos por consola.

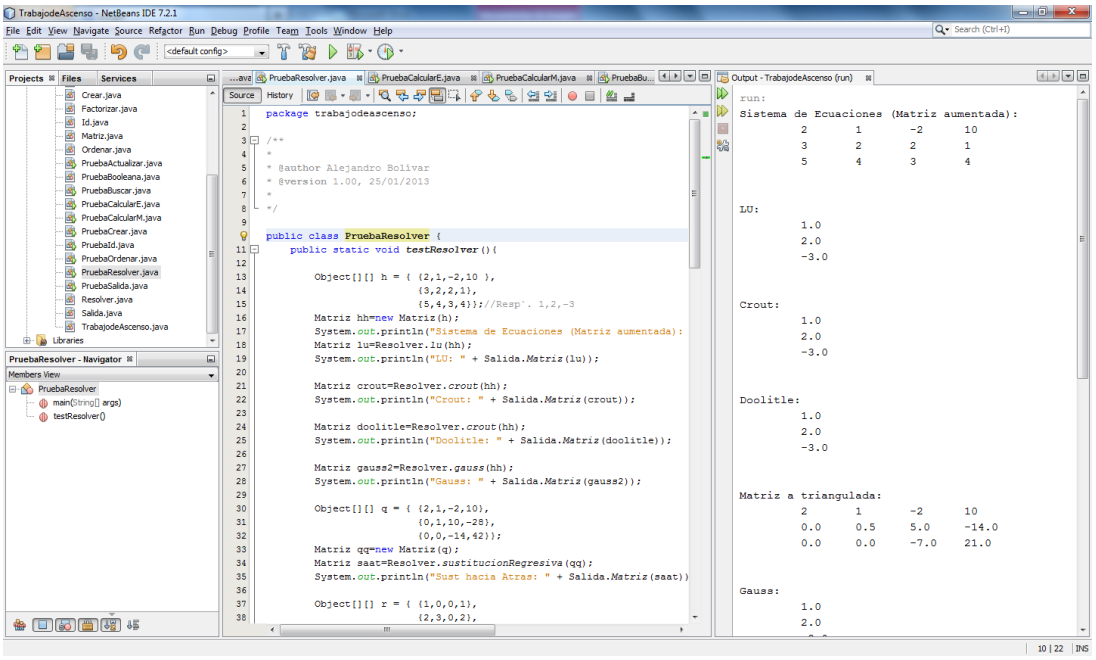


Figura 8. Ejecución de los métodos para resolver sistemas de ecuaciones lineales.

Búsqueda en arreglos bidimensionales

La implementación de métodos de búsqueda (ver anexo F.9) de datos en computación es muy usual cuando se maneja un gran volumen de datos, en esta biblioteca los métodos de búsqueda secuencial y binario, reciben como parámetros: la dimensión por la cual se realizará la búsqueda, es decir por fila o columna; el índice de la dimensión, y los índices entre los cuales se buscará el valor. El método de búsqueda alrededor de una celda, es un método recursivo que muestra la posición en la cual se encuentra el valor buscado en la matriz y la búsqueda de una submatriz en una matriz, devuelve la celda a partir de la cual se encuentra la submatriz.

`public static void alrededor(Matriz a, int x, int y)`, este método permite mostrar la fila y columna de la celda en la que se encuentra el valor buscado alrededor de la celda ubicada en la posición definida por los

parámetros *x*, *y*. Es un método recursivo ya que cada vez que consigue el valor buscado, el método nuevamente busca alrededor de dichos valores hasta que no consiga más coincidencias. La matriz *a* es la matriz en la cual se va a buscar el valor, y los argumentos *x*, *y*, son la posición de la celda alrededor de la cual se comenzará la búsqueda del valor.

`public static int binaria(Matriz a, char foc, int pos, int inicio, int fin, Object dato)`, la búsqueda binaria consiste en dividir el intervalo de búsqueda en dos partes, comparando el elemento buscado con el central. En caso de no ser iguales se redefinen los extremos del intervalo (según el elemento central sea mayor o menor que el buscado) disminuyendo el espacio de búsqueda. El proceso concluye cuando el elemento es encontrado, o bien cuando el intervalo de búsqueda se anula. Este método funciona únicamente para arreglos ordenados. Con cada iteración del método el espacio de búsqueda se reduce a la mitad, por lo tanto el número de comparaciones a realizar disminuye notablemente. Esta disminución resulta significativa cuanto más grande sea el tamaño del arreglo.

La búsqueda secuencial se realiza comparando cada uno de los valores en orden desde el primer índice identificado por el argumento inicio, hasta el último índice identificado con el argumento fin. Sí uno de los valores coincide con el valor que se busca entonces el método devuelve la posición del elemento buscado, por lo contrario sí el valor no se encuentra en el arreglo, entonces el método devolverá el valor de -1.

`public static int secuencial(Matriz a, char foc, int pos, int inicio, int fin, Object dato)`, el método de búsqueda secuencial ubica un elemento en la fila o columna indicada, los datos pueden estar ordenados o no.

`public static int secuencialOrdenado(Matriz a, char foc, int pos, int inicio, int fin, Object dato, char`

`orden)`, el método secuencial ordenado agrega un argumento adicional para complementar el criterio de búsqueda, el mismo depende si el arreglo está ordenado ascendentemente o descendentemente; esto mejora la búsqueda ya que al pasar el valor buscado, de seguro el valor no se encontrará.

`public static void subMatriz(Matriz a, Matriz b, int x, int y, int f, int c)`, este método permite buscar una submatriz *b* dentro de la matriz *a*, a partir de la celda definida por los parámetros *x*, *y*; los parámetros *f*, *c* representan la cantidad de filas y de columnas respectivamente del arreglo *b*.

Prueba de los métodos de búsqueda.

En la figura 9, se muestra la ejecución de los métodos para realizar la búsqueda en matrices, inicialmente se creó una matriz y se mostró en la consola, posteriormente se realizaron búsquedas de tipo secuencial y binaria, realizando también comprobaciones si la matriz se encontraba ordenada para poder efectuar la búsqueda en el caso de secuencial ordenado y binaria.

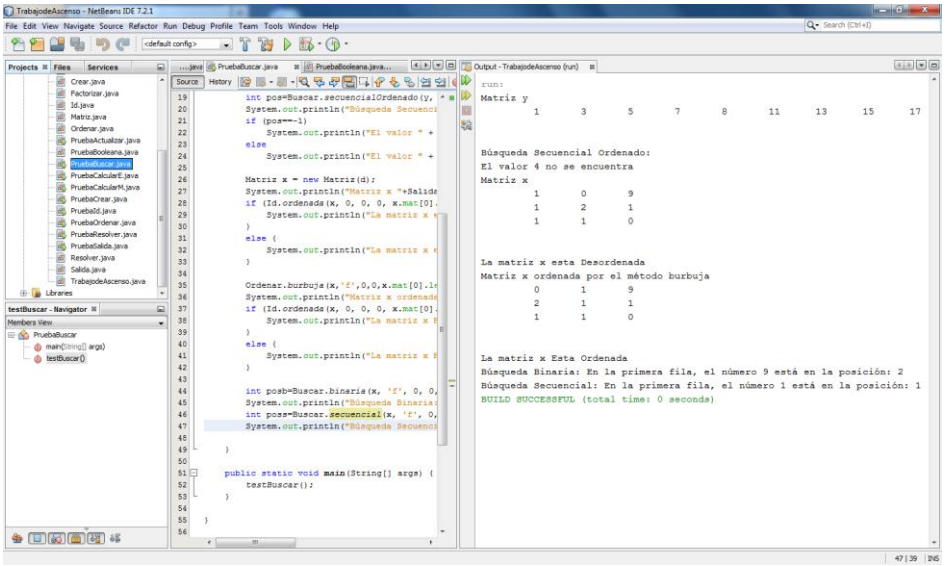


Figura 9. Ejecución de la clase buscar.

Ordenamiento de arreglo bidimensional

El ordenamiento de datos permite encontrar datos de manera más rápida para visualizar la información requerida. Los métodos de ordenamiento (ver anexo F.10) desarrollados en la clase permiten ordenar una fila o columna determinada, bien sea de manera dependiente o independiente del resto de filas o columnas y con un orden ascendente o descendente.

Los métodos de ordenamiento reciben como primer parámetro el arreglo que se ordenará, el parámetro *foc* es un carácter cuyos valores son *f* para ordenar por fila y *c* para ordenar por columna, el tercer parámetro *pos* define cual es el índice de la fila o columna definida en el parámetro anterior, los parámetros inicio y fin representan el índice complementario para definir el intervalo de valores que se ordenará en la fila o columna, el parámetro orden define mediante un carácter sí es ascendente se utiliza el carácter *a* y si es descendente el carácter *d*, y el último parámetro define si el ordenamiento es dependiente "true" o independiente "false" del resto de las filas o columnas. A continuación se muestra la sintaxis de cada uno de los métodos de ordenamiento desarrollados en la biblioteca.

```
    static void burbuja(Matriz a, char foc, int pos, int
inicio, int fin, char orden, boolean dependiente)

    static void insercion(Matriz a, char foc, int pos,
int inicio, int fin, char orden,boolean dependiente)

    static void quickSort(Matriz a, char foc, int pos,
int inicio, int fin, char orden, boolean dependiente)

    static void seleccion(Matriz a, char foc, int pos,
int inicio, int fin, char orden, boolean dependiente)
```

```
static void shellSort(Matriz a, char foc, int pos,
int inicio, int fin, char orden, boolean dependiente)
```

Prueba de los métodos de ordenamiento.

La prueba de los métodos de ordenamiento se llevó a cabo mediante la creación de una matriz *a*, leída desde un archivo de texto la cual es ordenada por los diferentes métodos. La matriz se ordenó en forma ascendente tomando como patrón la primera columna, también se definió el carácter dependiente de las columnas, por lo que todas las columnas se organizaron de acuerdo al orden de la primera columna. La ejecución del método mostró en consola los siguientes resultados (ver figura 10).

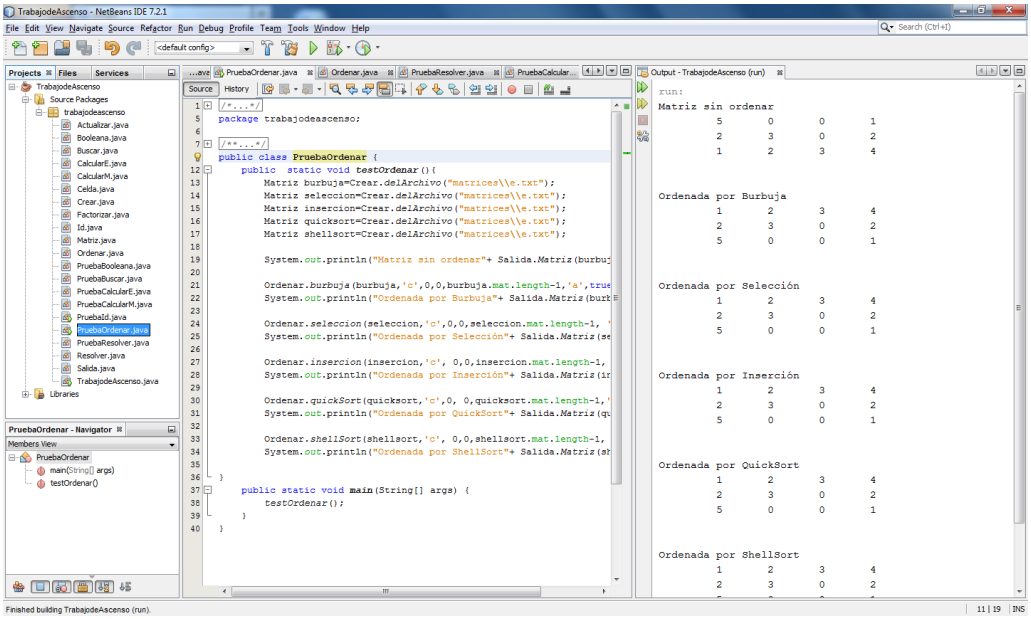


Figura 10. Ejecución de la clase ordenar.

Mostrar un arreglo

Los métodos de la clase Salida (ver anexo F.11), presentan por consola los elementos de una matriz. Se puede mostrar todas las celdas de la matriz, una parte de ella o guardar la matriz en un archivo.

Los siguientes métodos permiten mostrar toda la matriz por consola, estos métodos reciben la matriz como parámetro y devuelven una cadena de texto con los valores que contiene la matriz en cada celda

```
static String matriz(Matriz a)

static String matriz(double[][] a)

static String matriz(Object[][] a)
```

Los siguientes métodos muestran los elementos de la diagonal principal, la diagonal secundaria, el triángulo superior e inferior a la diagonal principal, y el triángulo superior e inferior de la diagonal secundaria de la matriz *a* por consola.

```
static String diagonalPrincipal(Matriz a)

static String diagonalSecundaria(Matriz a)

static String triangInfDiagPpal(Matriz a)

static String triangInfDiagSec(Matriz a)

static String triangSupDiagPpal(Matriz a)

static String triangSupDiagSec(Matriz a)
```

Para almacenar la matriz en un archivo se invoca el siguiente método, en el cual se indica el directorio y nombre del archivo y la matriz.

```
public static void guarda(String name, double a[][])
```

Prueba de la biblioteca para mostrar datos de una matriz.

En esta prueba se muestra por consola algunos de los elementos de una matriz, esto es útil cuando se desea enseñar al estudiante como realizar un recorrido determinado en la matriz o que reconozca ciertos sectores importantes de la matriz,

como lo son: las diagonales principal y secundaria o los elementos que conforman cada uno de los triángulos inferior y superior en la matriz con respecto a las diagonales, en la siguiente figura se visualizan los resultados de la prueba (ver figura 11).

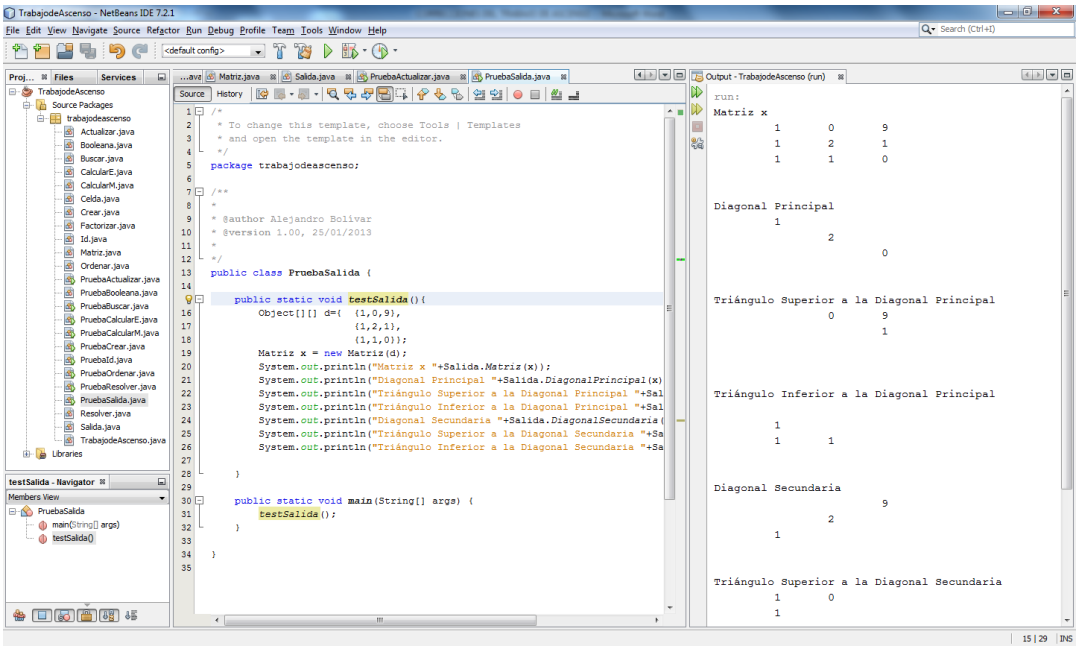


Figura 11. Ejecución de la clase salida.

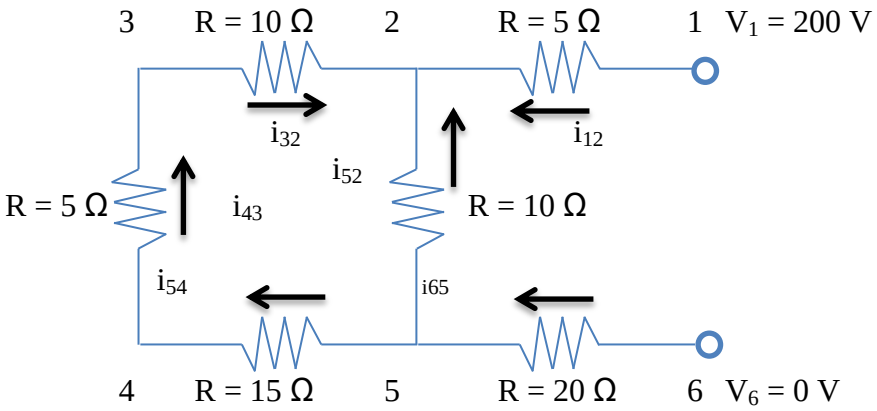
Aplicaciones prácticas en la ingeniería.

Ejemplo 1. Un problema común en la ingeniería eléctrica es la determinación de corrientes y voltajes en algunos puntos de los circuitos con resistores. Tales problemas se resuelven utilizando las reglas para corrientes y voltajes de Kirchhoff. La regla para las corrientes (o nodos) establece que la suma algebraica de todas las corrientes que entran a un nodo debe ser cero, donde todas las corrientes que entran al nodo se consideran de signo positivo. La regla de las corrientes es una aplicación del principio de conservación de la carga (Chapra & Canale, 2007).

Solución: La regla para los voltajes (o mallas) especifica que la suma algebraica de las diferencias de potencial en cualquier malla debe ser igual a cero. Para un

circuito con resistores, esto se expresa como: Σfem – ΣiR = 0, donde fem es la fuerza electromotriz de las fuentes de voltaje y R es la resistencia de cualquier resistor en la malla.

Resolver el siguiente circuito con resistores.



Según las direcciones supuestas para las corrientes, se aplica a cada nodo la regla de la corriente de Kirchhoff, obteniéndose:

$i_{12}+i_{52}+i_{32}=0$

$i_{65}-i_{52}-i_{54}=0$

$i_{43}-i_{32}=0$

$i_{54}-i_{43}=0$

Aplicando la regla de voltajes en cada una de las mallas se tiene:

$-i_{54}R_{54}-i_{43}R_{43}-i_{32}R_{32}+i_{52}R_{52}=0$

$-i_{65}R_{65}-i_{52}R_{52}+i_{12}R_{12}-200=0$

Sustituyendo los valores de las resistencias:

$-15i_{54}-5i_{43}-10i_{32}+10i_{52}=0$

$-20i_{65}-10i_{52}+5i_{12}-200=0$

Por lo tanto, el problema consiste en la solución del siguiente conjunto de seis ecuaciones con seis corrientes como incógnitas:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 10 & -10 & 0 & -15 & -5 \\ 5 & -10 & 0 & -20 & 0 & 0 \end{bmatrix} \begin{Bmatrix} i_{12} \\ i_{52} \\ i_{32} \\ i_{65} \\ i_{54} \\ i_{43} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 200 \end{Bmatrix}$$

Las corrientes son:

i_{12} = 6.1538, i_{52}=-4.6154, i_{32}=-1.5385, i_{65}=-6.1538, i_{54}=-1.5385, i_{43}=-1.5385

Los signos negativos en la corriente indican que el sentido es contrario al supuesto.

La figura 12 muestra la ejecución de la aplicación para resolver el circuito.

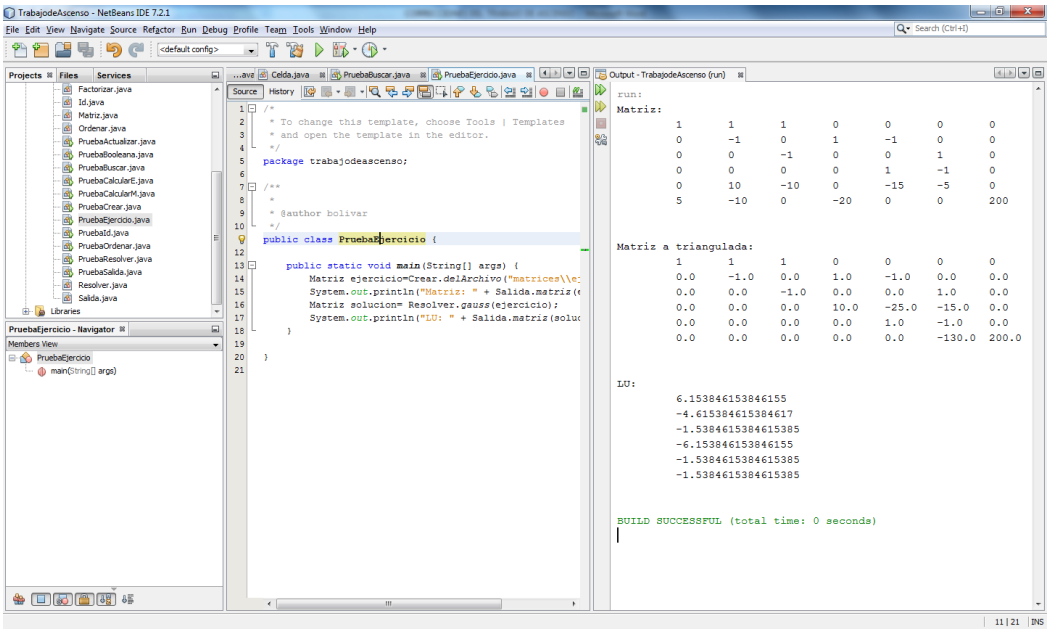


Figura 12. Ejecución de la aplicación para resolver el circuito.

Ejemplo 2. Un problema importante en la ingeniería estructural es encontrar las fuerzas y reacciones asociadas con una armadura estáticamente determinada. En la siguiente armadura (ver figura 13), las fuerzas (F) representan ya sea la tensión o la compresión sobre los componentes de la armadura. Las reacciones externas (H2, V2 y V3) son fuerzas que caracterizan cómo interactúa dicha estructura con la superficie de soporte. El apoyo fijo en el nodo 2 puede transmitir fuerzas horizontales y verticales a la superficie, mientras que el apoyo móvil en el nodo 3 transmite sólo fuerzas verticales. Se observa que el efecto de la carga externa de 1000 lb se distribuye entre los componentes de la armadura (Chapra & Canale, 2007).

Solución: Se realiza el diagrama de fuerza de cuerpo libre para cada nodo. La suma de las fuerzas en la dirección vertical y horizontal, deben ser cero para cada nodo, ya que el sistema está en reposo. Por lo tanto para el nodo 1,

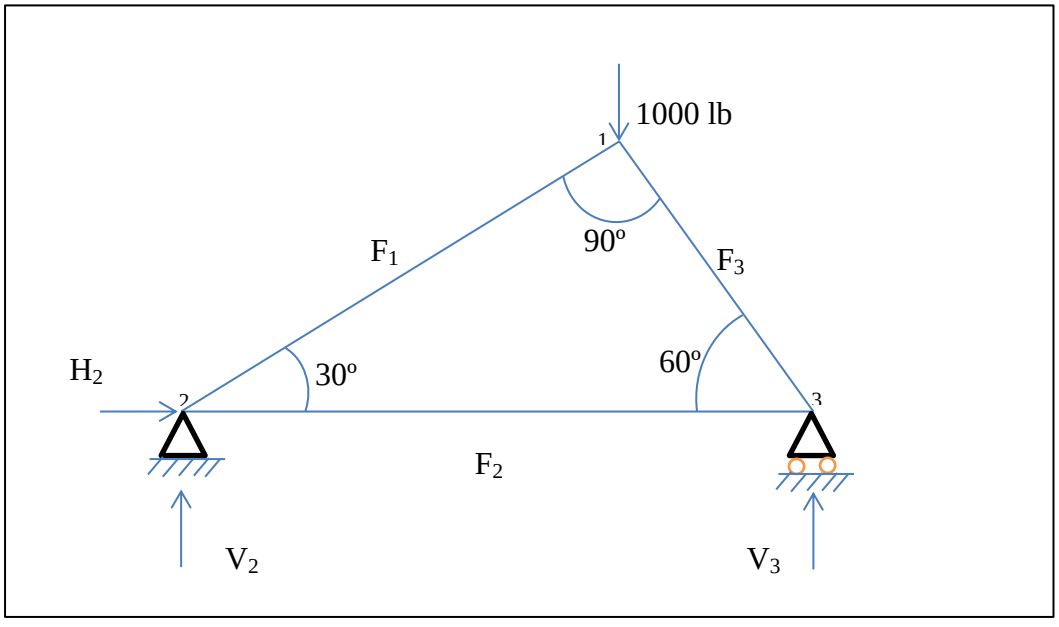


Figura 13. Fuerzas en una armadura estáticamente indeterminada.

$\Sigma FH = 0 = -F_1 \cos 30^o + F_3 \cos 60^o + F_{1,h}$

$\Sigma FV = 0 = -F_1 \text{ sen } 30^o - F_3 \text{ sen } 60^o + F1,v$

Para el nodo 2,

$\Sigma FH = 0 = F_2 + F_1 \cos 30^o + F_{2,h} + H_2$

$\Sigma FV = 0 = F_1 \text{ sen } 30^o + F_{2v} + V_2$

Para el nodo 3,

$\Sigma FH = 0 = -F_2 - F_3 \cos 60^o + F_{3,h}$

$\Sigma FV = 0 = F_3 \text{ sen } 60^o + F_{3v} + V_3$

Este problema se plantea como el siguiente sistema de ecuaciones de seis ecuaciones con seis incógnitas.

$$\begin{bmatrix} 0.866 & 0 & -0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 & 0 & 0 \\ -0.866 & -1 & 0 & -1 & 0 & 0 \\ -0.5 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0.5 & 0 & 0 & 0 \\ 5 & 0 & -0.866 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ H_2 \\ V_2 \\ V_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -1000 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

Resolviendo el sistema con una estrategia de pivote, se obtiene el siguiente resultado:

F_1=-500, F_2=433, F_3=-866, H_2=0, V_2=250, V_3=750

La matriz inversa es:

$$\begin{bmatrix} 0.866 & 0.5 & 0 & 0 & 0 & 0 \\ 0.25 & -0.433 & 0 & 0 & 1 & 0 \\ -0.5 & 0.866 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ -0.433 & -0.25 & 0 & -1 & 0 & 0 \\ 0.433 & -0.75 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Debido a que las fuerza externas no tienen efecto sobre la descomposición LU, no se necesita aplicar el método una y otra vez para estudiar el efecto de varias fuerzas externas sobre la armadura. Por ejemplo, para estudiar el efecto de fuerza

horizontales inducidas por un viento que sopla de izquierda a derecha. Si la fuerza del viento se puede idealizar como dos fuerzas puntuales de 1000 libras sobre los nodos 1 y 2, el vector de términos independientes es:

F^T = [-1000 0 1000 0 0 0]

Realizando el producto de la matriz inversa por el vector de términos independientes o resolviendo el sistema original aumentada con el nuevo vector, se obtiene los siguientes resultados:

F_1=866, F_2=250, F_3=-500

H_2=-2000, V_2=-433, V_3=433

Para un viento de la derecha, se tiene los siguientes términos independientes:

F^T = [-1000 0 0 0 -1000 0]

Resolviendo de la misma manera, se obtienen los siguientes resultados:

F_1=-866, F_2=-1250, F_3=-500

H_2=2000, V_2=433, V_3=-433

De esta manera la biblioteca puede favorecer el desarrollo de software educativo para la enseñanza y aprendizaje en la resolución de ejercicios, desarrollar aplicaciones para la simulación de casos prácticos en la ingeniería.

En la figura 14 muestra la ejecución de la aplicación para realizar el análisis de una armadura estáticamente determinada.

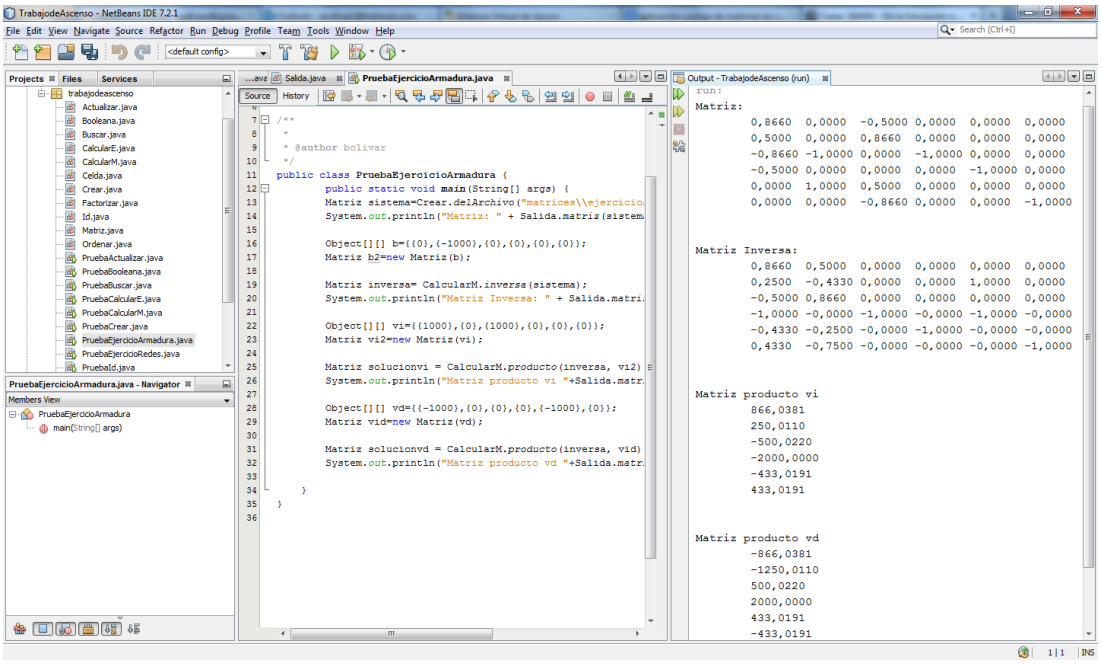


Figura 14 Ejecución de la aplicación para resolver el sistema de la armadura.

CAPÍTULO V

CONCLUSIONES

En los que respecta a la creación y desarrollo de los métodos para conformar la biblioteca se puede concluir que:

Los métodos desarrollados en este trabajo, son una base de partida para el desarrollo de nuevos métodos que contribuyan a la resolución de problemas en el área de computación e ingeniería mediante la implementación de matrices, también ofrecen la posibilidad de mejorar la eficiencia de los métodos desarrollados.

La posibilidad de tener el código fuente a la mano, facilita no solo lo indicado anteriormente, sino que también ofrece la posibilidad de poder desarrollar software educativo tanto para la enseñanza como para el aprendizaje, en el cual se pueda estudiar el proceso de solución o de trabajo con la matriz paso a paso. El hecho de que la herramienta pueda interactuar con el usuario mientras realiza las operaciones en la matriz, ya es una actividad educativa que puede contribuir con la enseñanza y el aprendizaje del manejo de matrices. Adicionalmente, se puede desarrollar simuladores en las asignaturas que así lo requieran para realizar análisis más abstractos que en una clase convencional se dificulta desarrollar.

El tiempo para el aprendizaje y desarrollo de nuevas aplicaciones tomando como base esta biblioteca disminuye, debido a que la biblioteca está conformada por los métodos más utilizados en la resolución de problemas matriciales y en el área de computación.

La biblioteca también es muy importante para el desarrollo de programas en el área de ingeniería y de computación para trabajos de grado o de investigación ya que inclusive con esta fase inicial, presenta una gran cantidad de aplicaciones lo que

favorece el tiempo de edición del código para programas más especializados, en los que interese el resultado final de las operaciones.

Finalmente la biblioteca ofrece otra posibilidad, que es la de poder realizar comparaciones entre métodos para poder realizar análisis más profundos que en una clase convencional no sería posible poder desarrollar, tal es así el caso por ejemplo de los métodos de búsqueda, de ordenamiento o la aplicación de los diferentes métodos de factorización de sistemas de ecuaciones lineales.

RECOMENDACIONES

Esta biblioteca se puede utilizar como base para el desarrollo de software educativo y simuladores en diferentes áreas como: álgebra lineal, ecuaciones diferenciales, computación, manejo de fluidos, circuitos eléctricos, sistemas de transporte, métodos numéricos entre otras.

Para la creación de proyectos de investigación es una gran ventaja ya que contiene una gran cantidad de métodos de amplio uso para el cálculo numérico de aplicaciones en la ingeniería y operaciones en computación.

Este trabajo se debe complementar con el desarrollo de otros métodos de aproximación numérica: interpolación, diferenciación, integración, raíces y otros; operaciones con matrices densas y esparcidas, programación paralela. El potencial de la biblioteca se beneficia con el mayor alcance que pueda tener en cuanto a las áreas de aplicación.

Incorporar como asignaturas obligatorias en el pensum de Ingeniería Civil, las asignaturas Computación II y Computación Avanzada, para el mejoramiento de la formación tecnológica del estudiante en el área de programación con respecto al uso de matrices y en la programación orientada a objetos, ya que en su área tiene un amplio uso el empleo de las matrices para la solución de problemas.

REFERENCIAS

Referencias Bibliográficas

Ley de Universidades. (1970). Caracas, Venezuela: Gaceta Oficial de la República Bolivariana.

Constitución de la República Bolivariana de Venezuela. (30 de Diciembre de 1999). Caracas, Venezuela: Gaceta Oficial Nº36860.

(2006). *Manual de trabajos de grado de especialización y maestria y tesis doctorales.* Universidad Pedagógica Experimental Libertador, Caracas.

Almeida Benítez, P. R., Márquez Rodríguez, I., & Franco Brañas, J. R. (2001). Una aplicación del cálculo matricial a un problema de ingeniería. *Divulgaciones Matemáticas, 9*(2), 197-205.

Álvarez, M., Molinás, P., Steegmann, C., Martínez, F., & Juan, Á. A. (2002). MathCad y la UOC una experiencia en E-Learning. *Fórum Tecnológico*, 4-6.

Arias, F. (2006). *El proyecto de investigación* (Quinta ed.). Caracas, Venezuela: Episteme.

Arocha Correa, C. C., & López Hernández, M. L. (2000). *Aprendizaje para realizar un estudio de mercado.* Universidad de Carabobo.

Bayona Cardenas, V. M. (2010). *Scribd.* Recuperado el 27 de 04 de 2012, de http://es.scribd.com/doc/34843241/Solucion-de-Sistemas-de-Ecuaciones-Lineales-Metodos-Directos-e-Iterativos

Burden, R. L., & Faires, J. D. (2002). *Análisis Numérico* (Séptima ed.). (Cengage Learning Editores, Ed.) México: Ediciones Paraninfo, S.A.

Calderon Vilca, H. (2008). *Matemáticas discretas para la ciencia computación.* Puno, Perú: Pacífico.

Catalani, G., & Gutierrez, J. (26 de Enero de 2006). *Reporte de la búsqueda de bibliotecas numéricas de álgebra lineal, de alto rendimeinto, de dominio público en internet.* Mérida.

Chapra, S. C., & Canale, R. P. (2007). *Métodos numéricos para ingenieros.* México: McGraw-Hill/Interamericana editores, S.A. de C.V.

Deitel, P. J., & Deitel, H. M. (2008). *Java cómo programar*. Pearson Educación de México, S.A. de C.V.

Di Mare, A. (2010). Introducción al uso de bibliotecas de algebra para estudiantes de ingeniería. *Eighth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2010).* Arequipa.

Grossman, E. I. (2008). *Algebra Lineal con Aplicaciones* (Sexta ed.). México: Mc Graw Hill Interamericana.

Hernández Sampieri, R., Fernandez Collado, C., & Baptista Lucio, P. (2006). *Metodología de la Investigación* (Cuarta ed.). México: Mc-Graw Hill.

Joyanes Aguilar, L. (2008). *Fundamentos de programación.* Madrid: McGraw-Hill/Interamericana de españa, S. A. U.

Kincaid, D., & Cheney, W. (1994). *Análisis numérico.* Addison-Wesley Iberoamericana.

Lay, D. (1999). *Algebra Lineal y sus Aplicaciones* (Segunda ed.). México: Pearson Prentice Hall.

Lipschutz, S. (1992). *Algebra Lineal* (Segunda ed.). Madrid, España: Mcgraw Hill.

Ruiz de Rivillo, M. A., & Chacón, C. R. (1992). *Procesamiento de datos I.* Universidad Nacional Abierta.

Shoichiro, N. (1992). *Métodos numéricos aplicados con software.* México: Prentice-Hall Hispanoamericana, S.A.

Referencias Electrónicas.

Gurin, S. (2004). *TLDP-ES/LuCAS: servicios editoriales para la documentación libre en español*. Recuperado el 10 de 10 de 2012, de http://es.tldp.org/Tutoriales/doc-programacion-algoritmos-ordenacion/alg_orden.pdf

TIOBE. (26 de 03 de 2012). *TIOBE*. Obtenido de http://www.tiobe.com/content/paperinfo/tpci/index.html

Bilblioteca JAMA. http://math.nist.gov/javanumerics/jama/

Biblioteca la4j: http://code.google.com/p/la4j/

Biblioteca Colt. http://acs.lbl.gov/software/colt/

Control de Estudios de la Facultad de Ingeniería de la Universidad de Carabobo. http://dae.ing.uc.edu.ve/index.php?mod=pagina&node=62

ANEXO.

Anexo A. Operacionalización de las variables

Objetivo General: Desarrollar una biblioteca de clases en java, para resolver problemas de computación y álgebra matricial, mediante el uso de arreglos bidimensionales.				
Objetivo específico	**Dimensiones**	**Definición**	**Indicadores**	**Ítems Asociados**
Diagnosticar, mediante un proceso de levantamiento de información, las necesidades esenciales de implementación de matrices en la resolución de problemas, en el área de computación y de ingeniería	Aptitudes Estudiantiles.	Proceso para determinar las necesidades de los alumnos, relacionado con la utilización de bibliotecas en Java, para la resolución de problemas mediante el álgebra matricial	• Percepción del ámbito real	1, 2, 3, 4, 5

Fuente: Propia

Anexo B. Operacionalización de las variables (Continuación)

Objetivo General: Desarrollar una biblioteca de clases en java, para resolver problemas de computación y álgebra matricial, mediante el uso de arreglos bidimensionales.				
Objetivo	**Dimensiones**	**Definición**	**Indicadores**	**Ítems Asociados**
Diseñar mediante un modelo orientado a objetos, la estructura, clases y métodos de la biblioteca para la creación, identificación, cálculos y operaciones matriciales que se emplean en el área de computación y de ingeniería	Pedagógica Científico-Tecnológica	Proceso para diseñar la propuesta de creación de la biblioteca, para implementarlo en la resolución de problemas de ingeniería y computación mediante el álgebra matricial	• Hábitos de estudio. • Desarrollo de expectativas académicas • Conocimiento sobre los contenidos conceptuales en las áreas de la especialidad.	6, 7, 8, 9, 10, 11, 12, 13

Fuente: Propia

Anexo C. Cuestionario dirigido al estudiante

Universidad de Carabobo
Facultad de Ingeniería
Departamento de Computación

Estimado Estudiante:

Ha sido seleccionado entre los miembros académicos de la facultad de ingeniería de la Universidad de Carabobo, para responder este cuestionario con fines académicos relacionados a la resolución de problemas mediante el álgebra matricial. Sus respuestas a este cuestionario constituyen una valiosa colaboración al desarrollo de mi trabajo de investigación. Es necesario que al responder lo haga de manera precisa, clara y sincera consigo mismo, para garantizar la veracidad de la información aquí recopilada.

Reciba mi agradecimiento por su colaboración,

Prof. Alejandro E. Bolívar P.

Cuestionario.

Ítems.	***Totalmente en desacuerdo***	***En desacuerdo***	***Ni de acuerdo, ni en desacuerdo***	***De acuerdo***	***Totalmente de acuerdo***
1. Prefiero escribir totalmente el código, aunque ya este creado.					
2. Es conveniente disponer de subprogramas para crear una aplicación.					
3. Es conveniente descargar o copiar el código necesario de cualquier fuente en internet para crear una aplicación para un proyecto en el computador.					
4. En las asignaturas de la carrera, se requiere del uso del computador para resolver problemas.					
5. Las aplicaciones actuales se deben desarrollar para trabajar en cualquier sistema operativo y para móviles.					
6. Es útil disponer de algún subprograma para el ordenamiento de datos.					
7. Es necesario disponer de un subprograma para la resolución de sistemas de ecuaciones lineales.					
8. En ocasiones es necesario buscar algún valor en un arreglo.					
9. En el manejo de matrices es importante poder reconocer que tipo de matriz se está utilizando.					
10. Un sistema de ecuaciones lineales, se puede resolver mediante la implementación de cualquier método de resolución.					
11. Mi trabajo de grado, requerirá la resolución de problemas mediante álgebra matricial o búsquedas u ordenamientos.					
12. La facultad de ingeniería de la Universidad de Carabobo, cuenta con aplicaciones que permiten resolver problemas de álgebra matricial, búsqueda, ordenamiento y resolución de sistemas de ecuaciones lineales.					
13. Existe software comercial que realizan operaciones matriciales, ordenamiento, búsqueda y resolución de sistemas de ecuaciones lineales.					

Adicionalmente, carrera que cursa:

□Mecánica □Civil □Eléctrica □Telecomunicaciones □Industrial □Química

Gracias por su colaboración.

Anexo D. Formato de validación del instrumento, carta al experto

Universidad de Carabobo
Facultad de Ingeniería
Departamento de Computación

Estimado Profesor:

A continuación se presenta un instrumento que permitirá la recolección de información útil para el desarrollo de la investigación titulada "Desarrollo de una biblioteca de clases en java para resolver problemas de computación y álgebra matricial mediante el uso de arreglos bidimensionales".

Se ha diseñado un instrumento dirigido a profesores del área de computación y matemática, con el propósito de sustentar la determinación de la necesidad de la creación de una biblioteca de clases en java para la resolución de problemas en matemática y computación que requieran el empleo de arreglos bidimensionales.

Usted ha sido seleccionado para participar como experto en la validación del instrumento, lo cual constituye una valiosa colaboración al desarrollo de este trabajo de investigación.

El juicio referente a la validación del instrumento permitirá reconocer específicamente la congruencia entre los ítems y los objetivos de la investigación, y la claridad con que están redactados.

Reciba mi agradecimiento por su colaboración,

Prof. Alejandro E. Bolívar P.

Instrucciones:

Lea cuidadosamente el objetivo general de la investigación, el objetivo asociado con el instrumento, y los ítems del cuestionario, e indique con una tilde (√) la alternativa que se ajuste a su criterio (Si o no) para cada elemento evaluado en el formato de validación (congruencia y claridad).

Objetivo general:

Desarrollar una biblioteca de clases en java, para resolver problemas de computación y álgebra matricial, mediante el uso de arreglos bidimensionales.

Objetivo específicos:

Con la finalidad de lograr el alcance del objetivo propuesto, se plantean los siguientes objetivos específicos:

- Diagnosticar, mediante un proceso de levantamiento de información, las necesidades esenciales de implementación de matrices en la resolución de problemas, en el área de computación e ingeniería.
- Diseñar mediante un modelo orientado a objetos, la estructura, clases y métodos de la biblioteca para la creación, identificación, cálculos y operaciones matriciales que se emplean en el área de computación e ingeniería.

Anexo E. Formato de validación del instrumento para docentes.

Nombre y Apellido:

Ocupación:

Cargo:

Item	**Congruencia**		**Claridad**		**Observaciones**
	Si	No	Si	No	
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

Anexo E. Formato de validación del instrumento para docentes.

Nombre y Apellido: Bassam Asfur

Ocupación: Docente en Computación

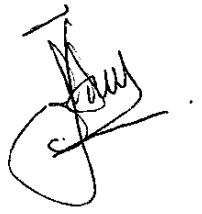
Cargo:

Item	Congruencia		Claridad		Observaciones
	Si	No	Si	No	
1	✓		✓		
2	✓		✓		
3	✓		✓		
4	✓		✓		
5	✓		✓		
6	✓		✓		
7	✓		✓		
8	✓		✓		
9	✓		✓		
10	✓		✓		
11	✓		✓		
12	✓		✓		
13	✓		✓		

Anexo E. Formato de validación del instrumento para docentes.

Nombre y Apellido: Francisco Duran.
Ocupación: Docente Computación
Cargo: Profesor Agregado.

Item	Congruencia		Claridad		Observaciones
	Si	No	Si	No	
1	X		X		
2	X		X		
3	X		X		
4	X		X		
5	X		X		
6	X		X		
7	X		X		
8	X		X		
9	X		X		
10	X		X		
11	X		X		
12	X		X		
13	X		X		



Anexo E. Formato de validación del instrumento para docentes.

Nombre y Apellido: Hernando González Acosta
Ocupación: Docente en el área Matemática
Cargo: Docente Agregado

Item	Congruencia		Claridad		Observaciones
	Si	No	Si	No	
1	✓		✓		
2	✓		✓		
3	✓		✓		
4	✓		✓		Muy general
5	✓		✓		
6	✓		✓		
7	✓		✓		
8	✓		✓		
9	✓		✓		
10	✓		✓		
11	✓		✓		
12	✓		✓		
13	✓		✓		

Anexo E. Formato de validación del instrumento para docentes.

Nombre y Apellido: ENRIQUE FLORES

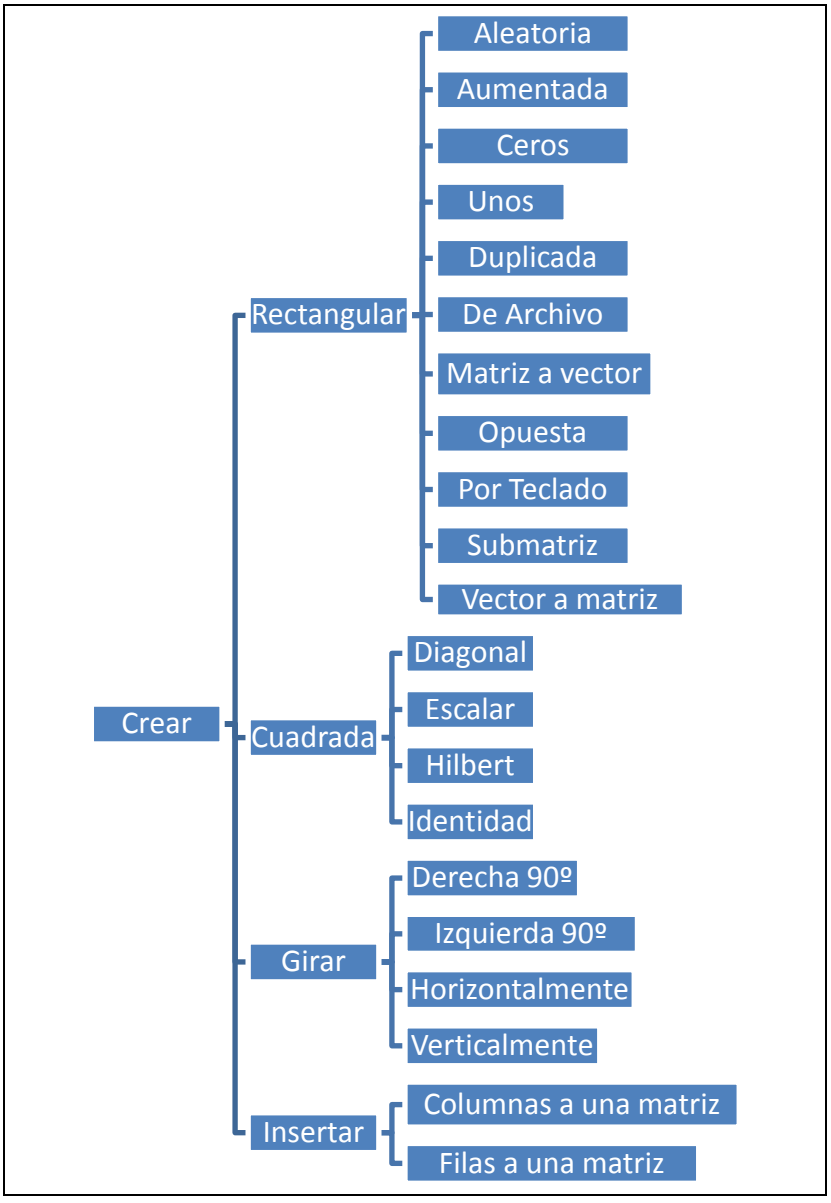
Ocupación: INGENIERO MECÁNICO

Cargo: DOCENTE DPTO. MATEMÁTICA - INVESTIGADOR DEL IMYCA.

Item	Congruencia		Claridad		Observaciones
	Si	No	Si	No	
1	X		X		
2	X		X		
3	X		X		
4	X		X		
5	X		X		
6	X		X		
7	X		X		
8	X		X		
9	X		X		
10	X		X		
11	X		X		
12	X		X		
13	X		X		

Anexo F. Estructura de las clases de la biblioteca.

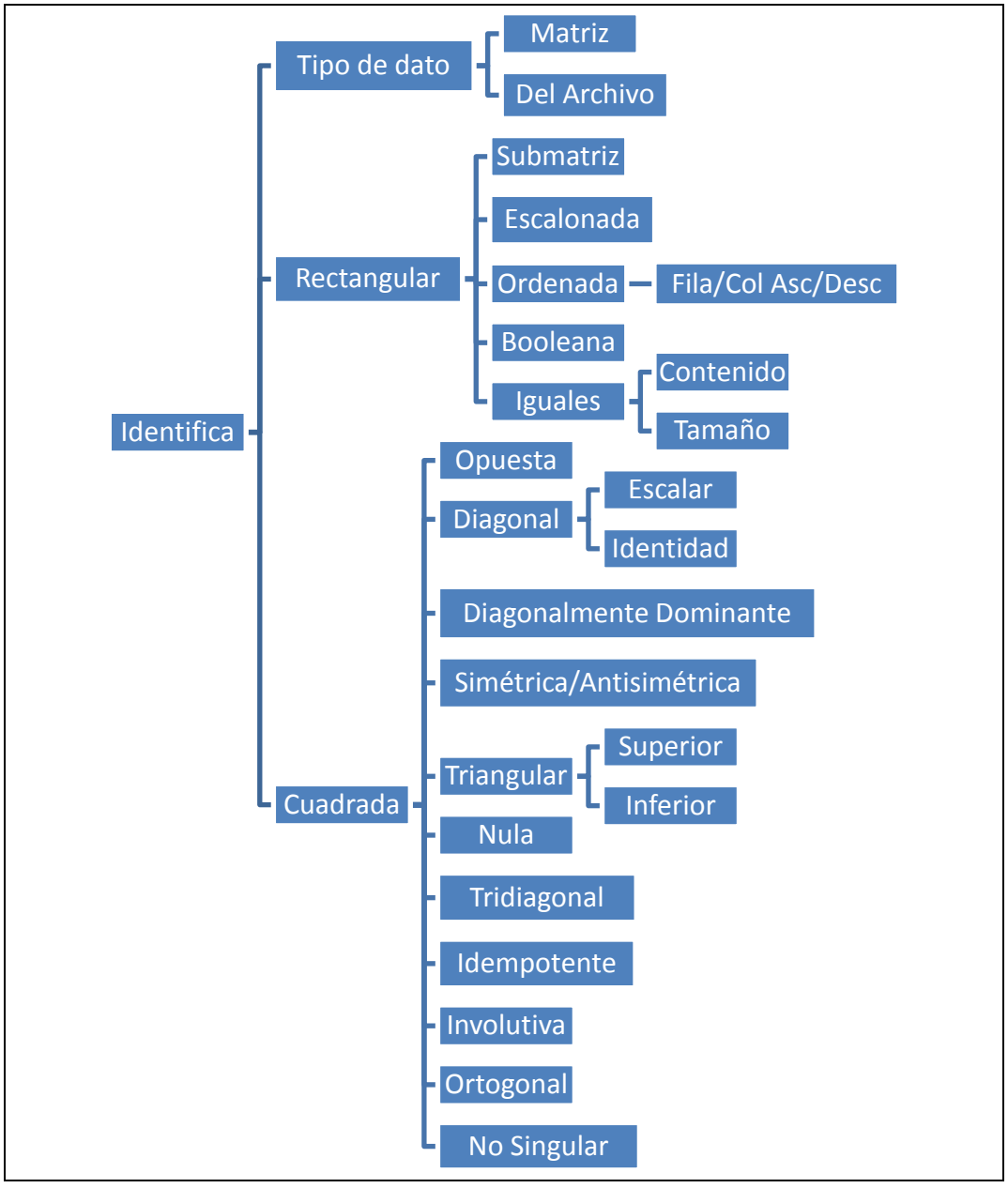
Anexo F.1 Estructura de la clase Crear.



Anexo F.1 Estructura de la clase Crear (continuación).

Crear
aleatoria(int cf,int cc,boolean e1,Object li,Object ls)
aumentada(Matriz a, Matriz x, boolean e1)
cantFilas(String ruta)
ceros(int cf, int cc, boolean e1)
ceros(Matriz a, boolean e1)
delArchivo(String ruta)
delTeclado(int cf, int cc)
diagonal(int n,Object k, Object v[], boolean e1)
dimensiones(String archivo)
duplicada (Matriz a)
eliminar(Matriz a,char foc, int k)
escalar(int n,Object k, boolean e1)
giraDerecha90(Matriz a)
giraIzquierda90(Matriz a)
girarH(Matriz a)
girarV(Matriz a)
hilbert(int n, boolean e1)
identidad(int n, boolean e1)
identidad(Matriz a, boolean e1)
insertarCMatriz(Matriz a, Object b[], int k)
insertarCMatriz(Matriz a, Matriz b, int columnainicio)
insertarFMatriz(Matriz a, Object b[], int k)
insertarFMatriz(Matriz a, Matriz b, int filainicio)
lee(String archivo)
matrizVector (Matriz a, int cf, int cc)
opuesta(Matriz a)
subMatriz (Matriz a,int fi,int ci, int cf, int cc)
subMatriz (Matriz a,Matriz b, int fi,int ci, int cf, int cc)
unos(int cf,int cc,boolean e1)
vectorMatriz (Matriz a, int cf, int cc)

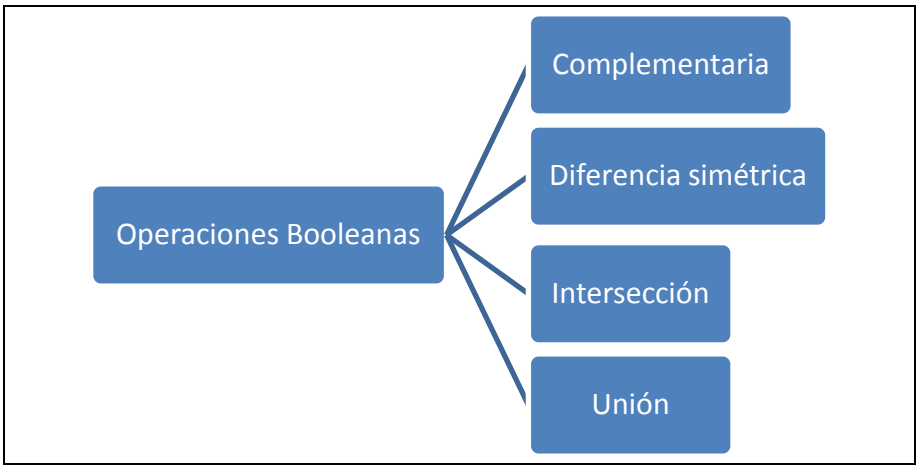
Anexo F.2. Estructura de la clase *Id*.



Anexo F.2. Estructura de la clase *Id*. (continuación)

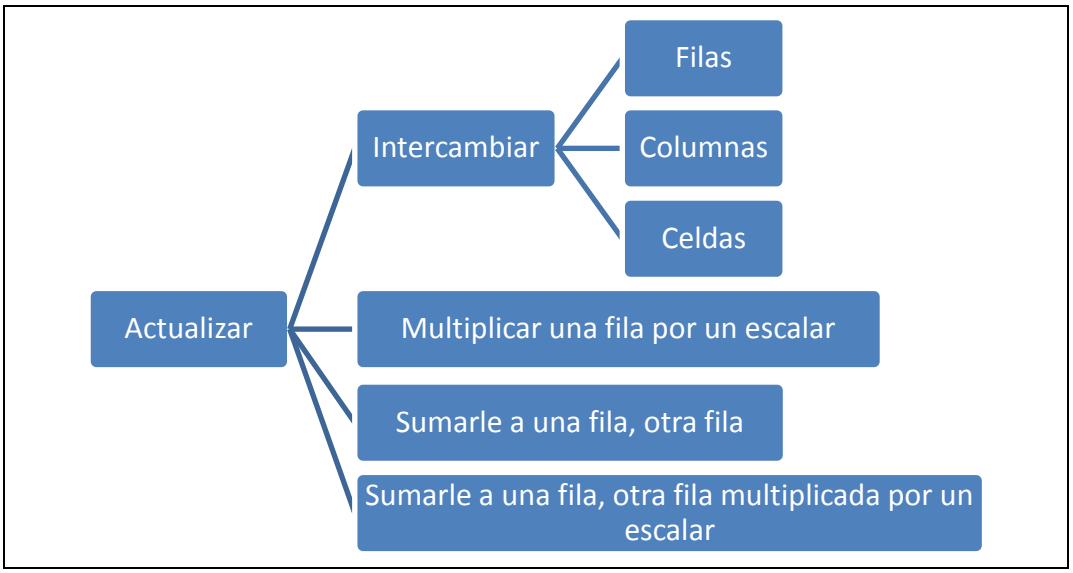
Id
antisimetrica(Matriz a)
booleana(Matriz a)
cuadrada(Matriz a)
definidaPositiva(Matriz a)
diagonal(Matriz a)
diagonalmenteDominante(Matriz a)
escalar(Matriz a)
escalonada(Matriz a)
esNoSingular (Matriz a)
idempotente(Matriz a)
identidad(Matriz a)
iguales(Matriz a, Matriz b)
igualTamaño(Matriz a, Matriz b)
involutiva(Matriz a)
nula(Matriz a)
opuesta(Matriz a,Matriz b)
ordenada(Matriz a, int foc, int pos, int inicio, int fin, int orden)
Ortogonal(Matriz a)
simetrica(Matriz a)
subMatriz (Matriz a, Matriz b, int x, int y)
tdMatrizArchivo(String ruta)
tipodeDato(Object z)
tipoEntero(Matriz a)
tipoEntero(Object z)
triangularInf(Matriz a)
triangularSup(Matriz a)
tridiagonal(Matriz a)

Anexo F.3. Estructura de la clase *Booleana*.



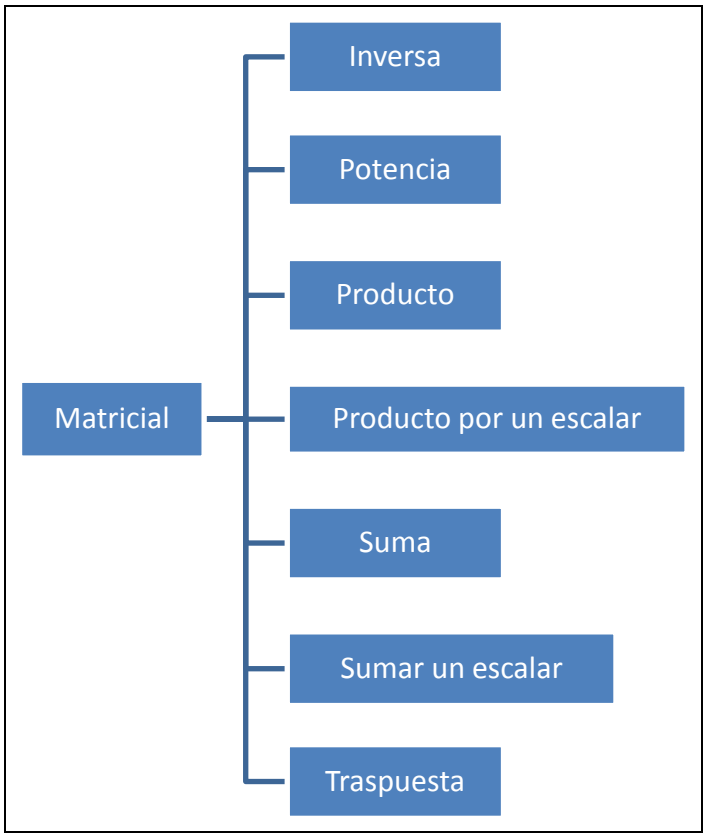
Booleana
complementaria(Matriz a)
diferenciaSimetrica(Matriz a,Matriz b)
interseccion(Matriz a, Matriz b)
union(Matriz a, Matriz b)

Anexo F.4. Estructura de la clase *Actualizar*.



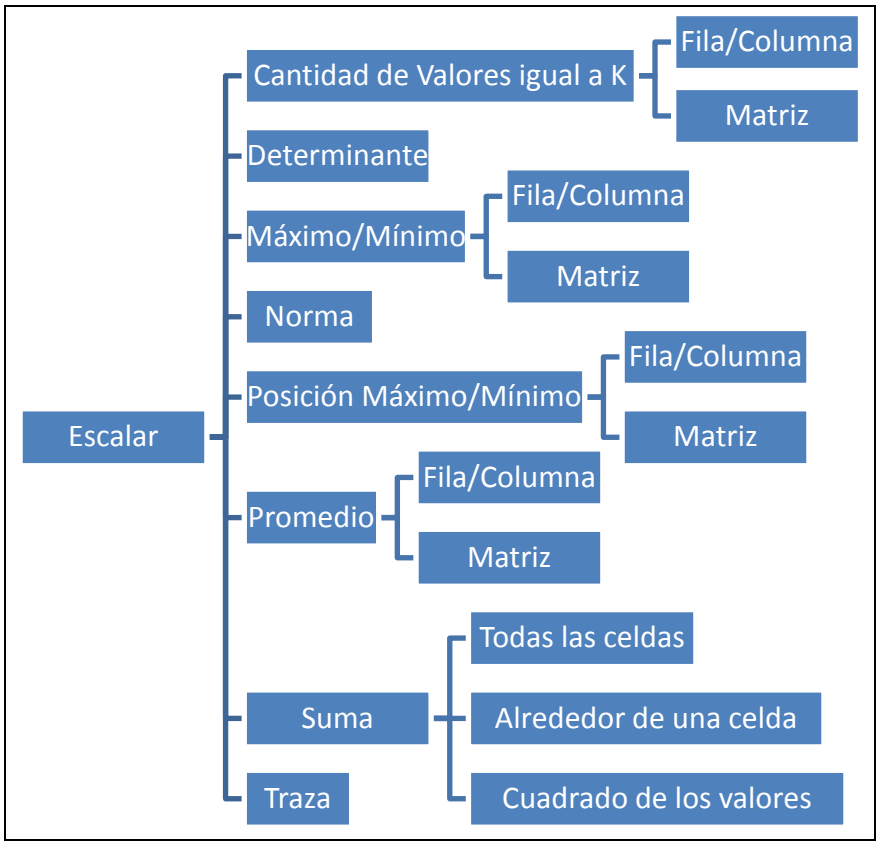
Actualizar
intercambiar(Matriz a, int f1, int c1, int f2, int c2)
intercambiar(Matriz a, char foc, int k, int l)
multiplicarFilaporK(Matriz a, Object k, int fila)
sumarFilas(Matriz a, int k, int l)
sumarFilas(Matriz a, int k, double c, int l)

Anexo F.5. Estructura de la clase *CalcularM*.



CalcularM
inversa(Matriz d)
potencia(Matriz a,Object k)
producto(Matriz a, Matriz b)
producto(Matriz a,Object k)
suma(Matriz a, Matriz b)
suma(Matriz a, Object k)
traspuesta(Matriz a)

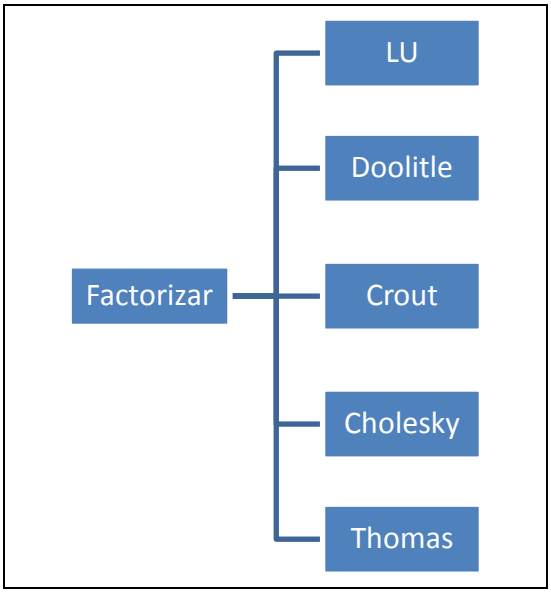
Anexo F.6. Estructura de la clase *CalcularE*.



Anexo F.6. Estructura de la clase *CalcularE* (continuación).

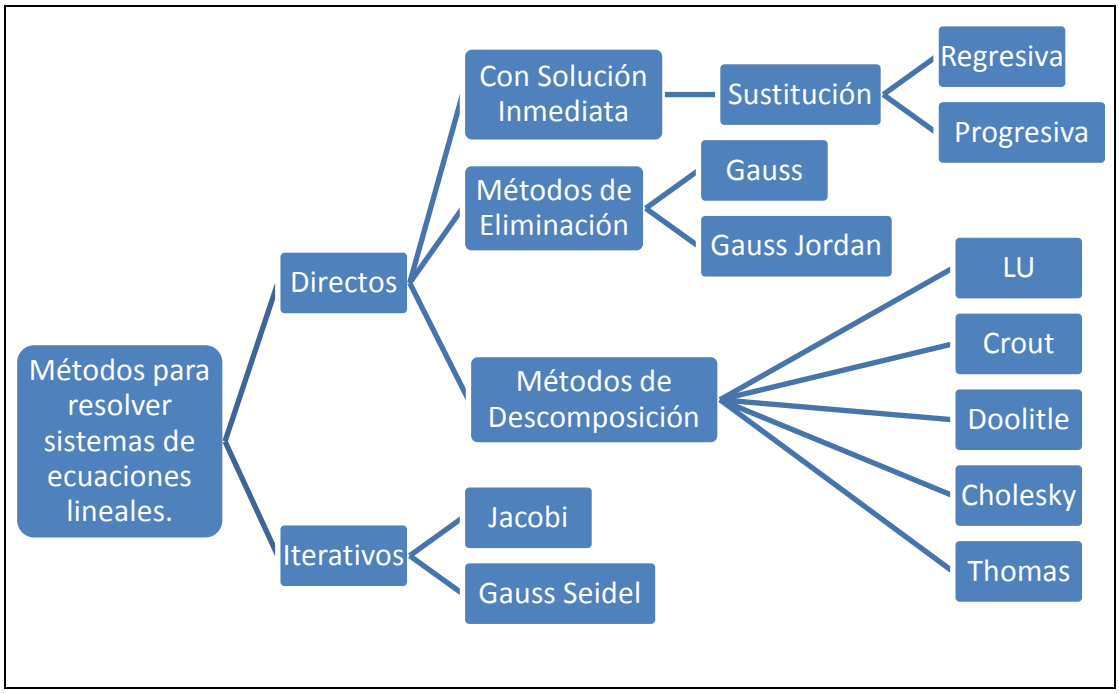
CalcularE
cantidad(Matriz a, char foc,int pos,Object k)
cantidad(Matriz a,Object k)
determinante(Matriz a)
maximo(Matriz a)
maximo(Matriz a, char foc, int pos)
minimo(Matriz a)
minimo(Matriz a, char foc, int pos)
norm(Matriz a)
posMaximo(Matriz a)
posMaximo(Matriz a, char foc, int pos)
posMinimo(Matriz a)
posMinimo(Matriz a, char foc, int pos)
promedio (Matriz a)
promedio (Matriz a,char foc, int pos)
suma (int k, int l,Matriz a)
suma (Matriz a)
suma(Matriz a,char foc, int pos)
sumaCuadrado(Matriz a)
traza (Matriz a)

Anexo F.7. Estructura de la clase *Factorizar*.



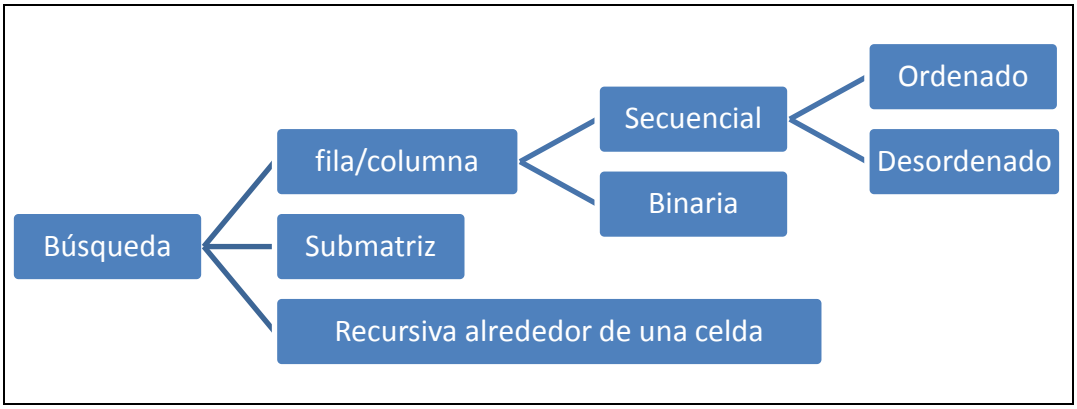
Factorizar
cholesky(Matriz a, Matriz l, Matriz u)
crout(Matriz a, Matriz l, Matriz u)
doolitle(Matriz a, Matriz l, Matriz u)
lu(Matriz a1,Matriz l, Matriz u)
thomas (Matriz a1,Matriz l, Matriz u)

Anexo F.8. Estructura de la clase *Resolver*.



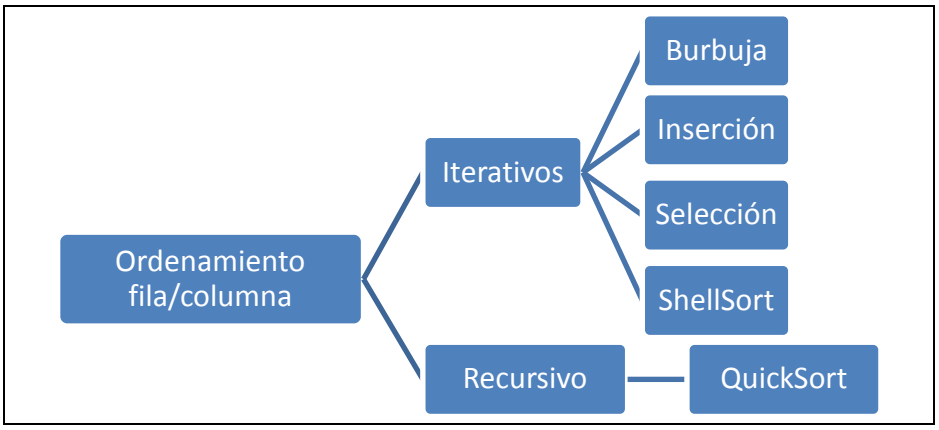
Resolver
static void cerosdebajoDiagonal (Matriz a, int f) static Matriz crout (Matriz a1) static Matriz cholesky (Matriz a1) static Matriz doolitle (Matriz a1) static Matriz eliminaciongaussiana(Matriz a) static Matriz eliminaciongaussjordan(Matriz a) static void jacobi(Matriz a, Matriz b, Matriz x0, Matriz x, float max) static Matriz lu (Matriz a1) static Matriz gauss(Matriz a) static Matriz gaussjordan(Matriz a) static void Pivoteo(Matriz a,int f) static Matriz Resolver(Matriz a, Matriz l, Matriz u) static Matriz sustitucionProgresiva(Matriz a) static Matriz sustitucionRegresiva(Matriz a) static Matriz thomas (Matriz a1)

Anexo F.9. Estructura de la clase *Buscar*.



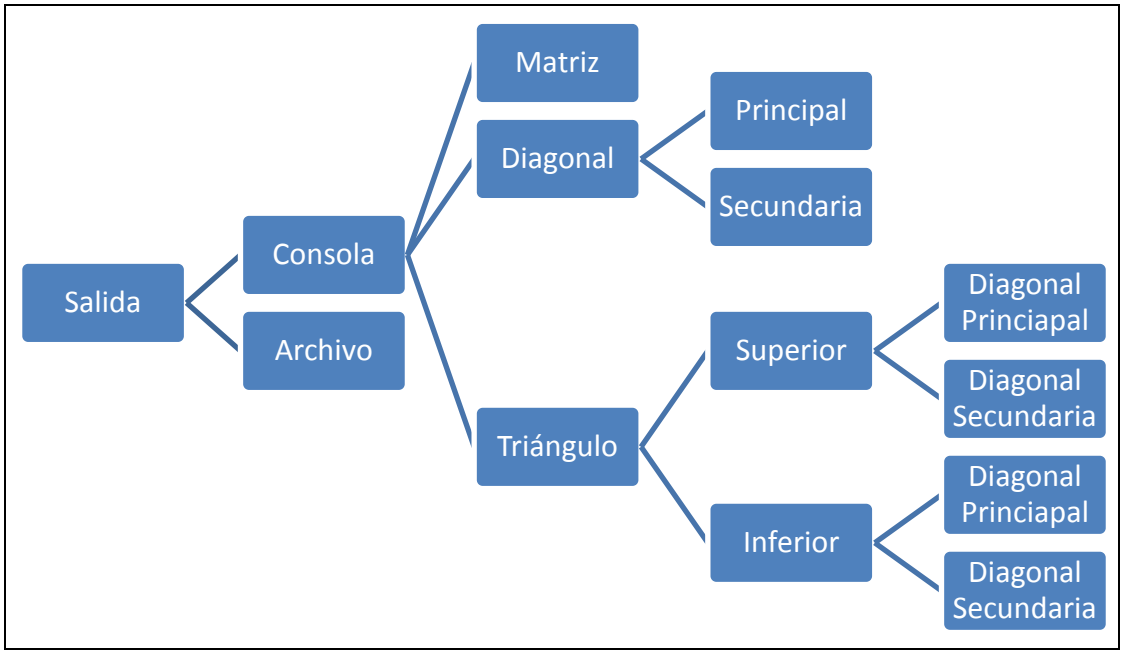
Buscar
alrededor(Matriz a, int x, int y, Object dato)
binaria(Matriz a, char foc, int pos, int inicio, int fin, Object dato)
secuencial(Matriz a, char foc, int pos, int inicio, int fin, Object dato)
secuencialOrdenado(Matriz a, char foc, int pos, int inicio, int fin, Object dato, char orden)
subMatriz(Matriz a, Matriz b, int x, int y, int f, int c)

Anexo F.10. Estructura de la clase *Ordenar*.



Ordenar
burbuja(Matriz a, char foc, int pos, int inicio, int fin,char orden, boolean dependiente)
insercion(Matriz a, char foc, int pos, int inicio, int fin, char orden, boolean dependiente)
quickSort(Matriz a,char foc,int pos, int inicio, int fin, char orden, boolean dependiente)
seleccion(Matriz a, char foc, int pos, int inicio, int fin, char orden, boolean dependiente)
shellSort(Matriz a, char foc, int pos, int inicio, int fin, char orden, boolean dependiente)

Anexo F.11. Estructura de la clase *Salida*.



Salida
diagonalPrincipal(Matriz a)
diagonalSecundaria(Matriz a)
guarda(String name, double a[][])
matriz(double[][] a)
matriz(Matriz a)
matriz(Object[][] a)
triangInfDiagPpal(Matriz a)
triangInfDiagSec(Matriz a)
triangSupDiagPpal(Matriz a)
triangSupDiagSec(Matriz a)